UNIVERSIDAD POLITÉCNICA DE MADRID

FACULTAD DE INFORMÁTICA

TRABAJO FIN DE CARRERA

DESARROLLO DE LA INTERACCIÓN EN EL NIVEL SOCIAL DE UNA ARQUITECTURA COGNITIVA PARA AGENTES

AUTOR: Carlos Herrero García

TUTOR: Ricardo Imbert Paredes

Mayo, 2009

A mis padres

AGRADECIMIENTOS

En este espacio quiero dar las gracias a todos y cada uno de aquellos que habéis hecho posible la realización de este proyecto. En primer lugar a mi tutor, Ricardo Imbert, por guiar e inspirar mi trabajo de la mejor manera posible, ofreciéndome siempre sus consejos y las ideas más originales. De igual manera quiero agradecerle a Alberto, mi compañero de proyecto, su apoyo y colaboración a lo largo de este año de trabajo.

Quiero agradecerles a mis padres la confianza que me han dado todos estos años de Ingeniería Informática, mostrándome siempre su apoyo frente a las dificultades y su alegría con los triunfos. También tengo presente a Luisa y a Toñi, sin las que no podría haber llegado hasta aquí. Y a Kelly, por ser siempre mi primera línea de defensa contra los sinsabores.

Además quiero daros las gracias a todos los compañeros del Decoroso que me habéis echado una mano: Ramón, Jesús, Javi, Álvaro, David, Dani, Leticia y Alberto; gracias por estar siempre allí. En especial quiero agradecerle a Jackeline, por su disposición a compartir sus ideas con la mejor actitud.

Finalmente quiero destacar a todos los que aquí no nombro, pero que habéis aportado vuestro granito de arena acompañándome en éste camino, gracias de corazón.

Índice general

Capítulo 1

INTRODUCCIÓN

La búsqueda de un sistema software que reproduzca el comportamiento humano ha sido una de las metas de la informática desde sus comienzos, concretamente dentro del campo de la Inteligencia Artificial.

Uno de los grandes avances en esta senda ha sido la aparición de un nuevo paradigma de desarrollo de software, la orientación a agentes, que permite abordar la creación de sistemas, desde un punto de vista en el que cada pieza de software es un elemento con autonomía, capacidad social, reactividad y proactividad. Esto permite construir los llamados Sistemas Multi-Agente en los que la interacción y coordinación entre los diversos componentes autónomos (agentes) lleva a cabo la tarea para la que fue propuesta el sistema.

No todas las tareas pueden ser resueltas mediante Sistemas Multi-Agente, pero el conjunto de las mismas incluye desde complejas simulaciones de entornos reales hasta asistentes virtuales para interacción con humanos.

Existen diversas arquitecturas propuestas para el desarrollo de estos sistemas. COGNITIVA [Imbert, 2005] es una de ellas y es la base de este proyecto, que consiste en el desarrollo de diversas mejoras sobre la versión inicial de la arquitectura, y fundamentalmente en establecer los mecanismos de interacción entre agentes que permitan desarrollar el nivel social de la misma.

En la versión inicial, desarrollada en los trabajos de [Leal, 2005, Molina, 2005] se construyeron los niveles reactivo y deliberativo, así como los mecanismos funcionales básicos de la arquitectura. Esta versión funciona correctamente en los entornos contextuales desarrollados, sin embargo, presenta varios problemas como la baja calidad del código, así como la presencia de varios componentes obsoletos. Además carece de algunos elementos propuestos por la arquitectura, de los que el más destacable es el nivel social.

Sin las posibilidades que proporciona este nivel, cada agente se comporta como un elemento aislado, que se enfrenta a sus metas contando únicamente con sus propias capacidades. En otras palabras, sin el nivel social no podemos hablar propiamente de un Sistema Multi-Agente.

Los objetivos de este proyecto son, en primer lugar, realizar una tarea de *Rediseño y Extensión* de la versión inicial de COGNITIVA que mejore la calidad del código, actualice sus componentes y desarrolle nuevas funcionalidades propuestas en los fundamentos teóricos de la arquitectura.

La segunda misión consiste en diseñar e implementar los mecanismos de *Interacción social* que establezcan las bases funcionales del nivel social de la arquitectura.

Finalmente, una vez mejorada la versión de la arquitectura e incorporada la interacción social, se probarán estos desarrollos por medio de dos contextualizaciones. Una de ellas completamente nueva, el entorno *Ghost Squadron*, y otra que consiste en una extensión del entorno de *Sabana 3D* que incluye interacción entre los *Agentes Cebra*.

Además, se proponen los objetivos transversales de aplicar rigurosamente las metodologías de Ingeniería del Software más adecuadas en cada fase, ya sea dentro del paradigma de los agentes o de la orientación a objetos, así como utilizar las mejores herramientas y tecnologías disponibles que puedan ser de utilidad en el desarrollo.

El trabajo fin de carrera se desarrolla a lo largo de 8 capítulos:

- En el capítulo 2, *Arquitectura Cognitiva,* se exponen los fundamentos y la base teórica de la arquitectura sobre la que se desarrollará el presente trabajo.
- El capítulo 3, *Rediseño y Extensión de la Arquitectura*, consiste en una recopilación de las mejoras realizadas sobre la versión inicial de COGNITIVA, así como de nuevas capacidades añadidas a los planteamientos originales.
- Seguidamente en el capítulo 4 se desarrolla la *Interacción Social*, el mecanismo para comunicar agentes en COGNITIVA que constituye la principal extensión a la arquitectura aportada por este proyecto.
- Una vez definidos los cambios funcionales en COGNITIVA, en el capítulo 5 se especifica la nueva *Concreción Contextual Ghost Squadron* que utiliza acciones sociales de comunicación entre agentes. Además, en la última sección se incluye una comparativa de tiempos y esfuerzos entre las concreciones anteriormente desarrolladas y la de *Ghost Squadron*

- Además, para completar el estudio de la interacción social entre agentes, se incluye el capítulo 6 *Otras Concreciones Contextuales de Carácter Social*, que desarrolla una extensión del escenario *Sabana 3D* para incluir solicitud de servicios de un agente a otro.

- En el capítulo 7 se presentaran las *Conclusiones* obtenidas, no sólo de los resultados de la experimentación, sino del desarrollo realizado, incluyendo lenguajes, metodologías y conocimiento sobre la propia arquitectura.

- Finalmente, en el capítulo 8 se enumeran las diversas *Líneas de Trabajo Futuro* propuestas para continuar el trabajo realizado.

- Adicionalmente, además de la *Bibliografía*, se incluyen tres anexos, de los cuales el A y el B son *Ficheros de Configuración de las Concreciones Contextuales*, mientras que el C incluye el *Dominio para el Planificador JSHOP2* del entorno Sabana.

Capítulo 2

ARQUITECTURA COGNITIVA

2.1. Descripción de la Arquitectura

En este capítulo se explicarán los fundamentos de la arquitectura COGNITIVA [Imbert, 2005]. Los modelos de agentes, como el aquí expuesto, se definen por lo general en tres componentes:

- **Sensores**, encargados de percibir el estado del entorno y los cambios en el mismo.
- **Módulo cognitivo**, cuya función es interpretar lo percibido y decidir cómo actuar.
- **Actuadores**, para ejecutar las acciones decididas.

COGNITIVA modeliza el módulo cognitivo, aunque también describe las relaciones e intercambios con los otros dos componentes. Una de las principales características de la presente arquitectura es que es un ejemplo de *Arquitectura Híbrida*, es decir, que combina capacidades reactivas y deliberativas. A su vez, proporciona comportamientos de tipo social. En la figura 2.1.1 tenemos una representación de cómo se combinan los tres componentes del agente junto con la representación del modelo arquitectónico multicapa de COGNITIVA. Cada capa de la arquitectura proporciona al agente un modo de comportamiento:

- **Capa reactiva**, para proporcionar respuestas inmediatas a eventos percibidos en el entorno.
- **Capa deliberativa**, encargada de generar planes para cumplir las metas y objetivos del agente.

- **Capa social**, cuya tarea es también generar planes, pero teniendo en cuenta la interacción con el resto de agentes a partir de sus capacidades.

Las tres capas del modelo, muy diferentes en su función, son sin embargo homogéneas en cuanto a su entrada y salida. La manera en que reciben los datos de los sensores y proporcionan la salida a los actuadores es la misma en los tres casos. Este diseño es conocido como arquitectura *horizontal*.

Debido a este tratamiento conjunto de la interacción con los actuadores, es necesario un componente que proporcione esta salida común procedente de las tres fuentes. Este componente es el *organizador*, cuya misión es establecer un orden de ejecución entre las posibles acciones a ejecutar y entregar al actuador, la que proceda en función a este orden. Dado que el componente tiene una lógica de decisión basada en criterios cognitivos (prioridades), se ha decidido incluir al organizador dentro del módulo cognitivo.

Así mismo, los mensajes procedentes de los sensores acerca de los cambios en el entorno han de ser procesados para que las capas del módulo cognitivo los entiendan. El encargado de este proceso de traducción e interpretación es el *intérprete* que, a partir de la información percibida, elabora unas estructuras llamadas *perceptos* que son las que maneja el módulo cognitivo. En síntesis, el intérprete se encarga de transformar las percepciones en perceptos, filtrar las percepciones traduciendo sólo las que son de interés para el agente, y encaminar los perceptos hacia los diferentes componentes de la arquitectura.

También, a modo de recopilación, se definen los principales procesos establecidos hasta el momento en la arquitectura:

- El intérprete analiza cualquier evento, cambio de estado e información del entorno y

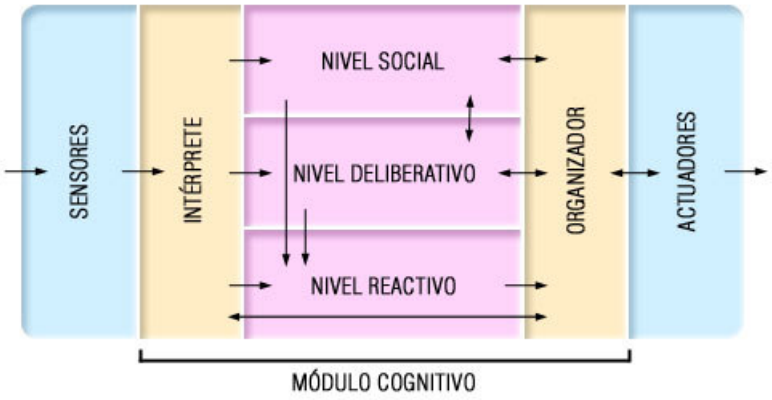

Figura 2.1.1: Arquitectura general del agente

entrega a cada componente de la arquitectura los que son relevantes para el mismo, previa traducción a perceptos.

- Los niveles deliberativo y social generan, a partir de lo entregado por el intérprete, nuevas metas. Además, proponen posibles planes para alcanzarlas, desde sus respectivos puntos de vista, y controlan las reacciones del nivel reactivo para hacerlas acordes con las nuevas metas establecidas.

- El nivel reactivo, a partir de lo entregado por el intérprete y del control que recibe de los niveles superiores, propone al organizador las acciones que considera necesarias.

- El organizador recoge las reacciones y planes elaborados por los distintos niveles, los organiza según la prioridad tratando de evitar conflictos y de maximizar el rendimiento, y manda a los actuadores aquellas que se puedan ejecutar en cada momento.

- Finalmente, el organizador comunica al intérprete las expectativas de las acciones que va entregando al actuador, y el intérprete señala al organizador si los perceptos afectan a las expectativas dadas.

Una vez descrito el flujo de información del módulo cognitivo, se explicarán las estructuras de conocimiento que forman los engranajes de la arquitectura. Estas estructuras o *creencias,* son la base del comportamiento emocional del agente; por tanto su mantenimiento y actualización van a ser tareas fundamentales del ciclo de vida del agente, así como la gestión de las relaciones entre ellas y su influencia sobre las acciones a realizar.

2.2. Creencias

Se denominan *creencias* a la información que el agente sabe o cree saber acerca del entorno, del resto de agentes, o del propio agente. Hablamos de creencias en lugar de hechos debido que la información que tiene el agente puede no ser completa, o incluso ser incorrecta; por tanto, las creencias son la información del mundo más probable para el agente. Esto, lejos de ser una desventaja, aporta alto grado de credibilidad al comportamiento del agente.

El tipo de creencias que maneja un agente es muy variado, dado que todo el conocimiento que atesore tiene que ser representable en forma de las mismas. Podemos hacer

una división en cuatro grupos, en la que tres (Recintos, Individuos y Objetos) representan creencias sobre los elementos tangibles del entorno y el último (Situación Personal) acerca de la información intangible sobre el entorno. En la figura 2.2.1 se representa una aproximación a una taxonomía de creencias en función del objeto de la creencia.

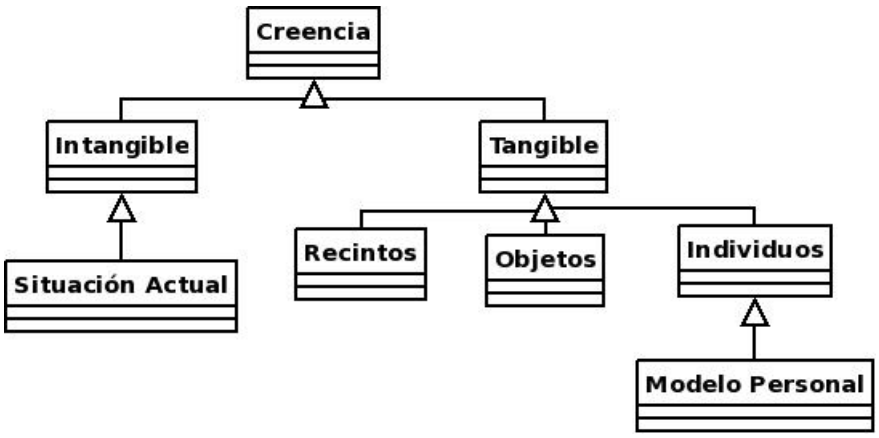

Figura 2.2.1: Creencias

2.2.1. Creencias acerca de Recintos

Los agentes manejados en COGNITIVA están situados en un entorno en el que se establecen recintos, de igual manera que los seres de un entorno real se mueven por lugares. Es sobre estos lugares o recintos sobre los que se definen las creencias explicadas en este apartado. La noción de recinto no tiene porque ser exclusiva de lugares *geométricos* sino que podemos estar hablando de estructuras lógicas por las que se mueve el agente.

En los ejemplos desarrollados se asume la representación externa corpórea del agente; este factor, junto con el planteamiento emocional de la arquitectura, hace que consideremos tres tipos de creencias acerca de recintos:

- **Características Definitorias de los recintos**. Son aquellos atributos relativos a los recintos que los identifican y definen sus características más estables, es decir, las que apenas varían en el tiempo. Algunos ejemplos son *identificación y posición.*

- **Estados Transitorios de los recintos**. Son atributos cuyo valor cambia de forma mucho más dinámica que las características definitorias. Dado este carácter cambiante, las creencias que el agente tenga acerca de estos atributos pueden no ser

correctas en un instante dado; así, puede ser necesario establecer un periodo de validez de las mismas. Ejemplos de estados transitorios son *luminosidad, número de objetos,...*

- **Actitudes hacia los recintos**. Este tipo de creencias forman parte de la información subjetiva que el agente tiene del recinto. La actitud hacia un recinto vendrá determinada en parte por las características y el estado en que se encuentra el recinto (los dos tipos anteriores); pero también estará establecida basándose en los sucesos acaecidos al agente en el propio recinto. Dado que son creencias personales del agente, su grado de certidumbre es muy alto.

2.2.2. Creencias acerca de Objetos

Así como en el entorno del agente hay recintos, también están presentes artefactos o instrumentos que determinan el comportamiento del agente. Hablaremos de ellos como objetos. Dada la variedad de entornos posibles, la clasificación de las creencias acerca de este tipo de objetos se establece de manera análoga a la de los recintos:

- **Características Definitorias de los objetos**. Al igual que los recintos, los objetos tienen una serie de atributos que permanecen estables a lo largo del tiempo. Son características muy dependientes del contexto, como el *color, identific*ación, *función...*

- **Estados Transitorios de los recintos**. Son atributos cuyo valor cambia de forma mucho más dinámica que las características definitorias. Ejemplos de estados transitorios son *última posición conocida, disponible o no,...*

- **Actitudes hacia los objetos**. Este tipo de creencias subjetivas hacia el objeto determinarán cómo de proclive es el agente a usar o no el mismo, o la forma de usarlo. La actitud hacia un objeto vendrá determinada en parte por las características y el estado en que se encuentra el objeto, y también a la satisfacción en el uso del objeto. Dado que son creencias personales del agente, su grado de certidumbre es muy alto.

2.2.3. Creencias acerca de Individuos. El Modelo Personal

Finalmente, además de creencias acerca de recintos y objetos, el agente mantiene creencias acerca del resto de individuos y, lo que es más importante, acerca de sí mismo. A

través de este conocimiento, el agente puede interactuar con el resto de individuos de la forma más adecuada, en función de sus intereses, o para lograr un bien común por medio de lo que se ha dado en llamar *inteligencia social*.

Además, como ya se ha recalcado, es fundamental para una arquitectura emocional el conocimiento del agente acerca de sí mismo. Estas creencias conforman el *modelo personal del agente*. Gracias a este conocimiento se puede conseguir que dos agentes del mismo tipo no se comporten igual ante el mismo evento e incluso el mismo agente reaccione de forma diferente ante una misma situación en dos instantes de tiempo diferentes. Se han clasificado las creencias acerca de individuos de igual manera que los anteriores tipos:

- **Características Definitorias de los individuos**. Todo individuo tiene unas características que lo definen y permanecen inmutables (o casi) en el tiempo. Aunque pueden establecerse tantos tipos de características definitorias como sea necesario, hay dos en concreto que deberían aparecer siempre en una arquitectura cognitiva: La *identificación* necesaria para que el individuo sea único y referenciable y, la más importante, los *rasgos de personalidad*, que en la presente arquitectura son los parámetros fundamentales que marcan el comportamiento del agente. Forman parte de su modelo personal y, al estar separados en diversos rasgos (en lugar de optar por *perfiles de personalidad*) ofrecen una mayor precisión y personalización de los comportamientos del agente.

- **Estados Transitorios de los individuos**. Son, de la misma forma que en los otros tipos de objetos, características que varían con cierta velocidad en el tiempo. En COGNITIVA se pueden definir tantos estados transitorios como sea necesario, pero de entre todos los posibles, destacan dos que son inherentes a la naturaleza emocional de la arquitectura: Las *emociones*, como podrían ser *miedo, sorpresa* o *alegría*, y los *estados físicos* como *sed, cansancio, hambre*... Estos dos tipos de creencias pertenecen así mismo al modelo personal del agente.

- **Actitudes hacia los individuos**. Las actitudes hacia otros individuos, más variables que las características definitorias, pero menos que los estados transitorios, serán determinantes a la hora de definir el comportamiento del individuo frente al resto de agentes. Cuando el agente va conociendo a otros individuos, establece una actitud preliminar en función de las características definitorias y sus estados transitorios, y la va modificando a lo largo de la interacción con dicho individuo.

2.2.4. Creencias acerca de la Situación Actual

Cuando nuestras creencias no se refieren a un objeto en concreto, sino a una característica intangible que se da en el entorno, estamos hablando de creencias sobre la situación actual. Este tipo de creencias está muy relacionado con la percepción de *estados* por parte del agente. En este proyecto se ha realizado una extensión sobre la arquitectura para implantar este tipo de creencias, antes no consideradas. Se han definido dos tipos de creencias sobre la situación actual, el *estado interno* del agente y el *estado de la comunicación* con otros agentes (ver en la sección 3.5).

2.3. Relaciones entre creencias

El factor clave de una arquitectura emocional es la influencia del estado de ánimo o emociones del agente sobre su comportamiento. Esto posibilita comportamientos que pueden parecer no del todo racionales, pero si muy coherentes con la realidad. En todo momento, los sucesos y cambios en el entorno, modificarán los valores de las emociones de los individuos y éstos influirán en elección de la siguiente acción a ejecutar, junto con el resto de creencias y otros elementos como las metas y reacciones (ver en la sección 2.6). Asumimos que el conjunto de emociones manejado por los individuos de un entorno es el mismo, precisamente para poder comparar el comportamiento de unos y otros en función de su estado anímico.

A su vez, las emociones de cada individuo estarán influenciadas por otros factores que no son puramente los hechos que suceden. Uno de estos factores van a ser los *rasgos de personalidad* del individuo, ya definidos en las características definitorias de los individuos. Si, en el mundo real, individuos con distinta personalidad tienen una respuesta emocional diferente ante los mismos hechos, aquí se trata de hacer una analogía por medio de la influencia de los rasgos de personalidad en las emociones. La personalidad influye sobre las emociones a partir de dos aspectos:

- El **modo** de influencia, la forma en que afecta un rasgo a una emoción. Puede ser aumentando su valor, disminuyéndolo, o haciendo otro tipo de operaciones más complejas.

- El **grado** de influencia, que es la fuerza con la que afecta, independientemente del modo.

Además de la influencia de los rasgos de personalidad, hay otras creencias del modelo personal que también tienen una cierta influencia en la respuesta emocional del agente. En este caso analizaremos la influencia que ejercen sobre las emociones otro tipo de estados transitorios, los e*stados físicos*. Este tipo de estado*s* transitorios vendrá dado en un conjunto común para todos los individuos, y sus valores cambiarán muy rápidamente. La influencia de los estados físicos se aplicará de la misma manera que la de la personalidad, es decir, con los conceptos de modo y grado de influencia. Cuando el valor de un estado físico cambie, los valores de la emociones influidas por él se verán modificados. A su vez, cuando el valor de una emoción cambie, este cambio se verá corregido por los estados físicos que influyan sobre esa emoción.

Puede parecer que las emociones son el único tipo de creencias que se ven influenciadas dentro de el modelo personal del agente, debido a que son las que tendrán influencia final sobre el comportamiento del agente. Sin embargo hay otro factor que se ha de considerar, la influencia de los rasgos de personalidad sobre las *actitudes* hacia los objetos, individuos o recintos. Esta influencia será la misma sea cuál sea el receptor de esa actitud, y actuará como un rectificador de los valores que establezca el diseñador para el objeto concreto. Por ejemplo, el rasgo *valor* (entendiendo por *valor* el grado de coraje o valentía innata del individuo) influenciará sobre la actitud *temor* sea cual sea el receptor de ese temor, por tanto será un corrector al valor de temor que establezca el diseñador. Esta influencia se producirá tanto en la creación de la actitud, como en su posterior modificación.

Para que esta información acumulada sobre las actitudes tenga su efecto sobre el comportamiento del agente es necesario finalmente considerar la influencia de las actitudes sobre las emociones. Esta influencia también será la misma independientemente de hacia quién va dirigida la actitud; de esta manera actuará como rectificador de las emociones. Por tanto un individuo que tenga una influencia entre el *temor* y el *miedo* muy alta, sufrirá que cuando otro individuo que le cause temor aparezca en el entorno, el miedo se le dispare a niveles altos. Esta influencia tendrá lugar tanto cuando se modifica el valor de una emoción (rectificando con las actitudes que la influencian) como de una actitud (influenciando sobre las emociones relacionadas)

Tanto la influencia de la personalidad sobre la actitud, como la de ésta sobre las emociones, seguirá el mismo cálculo ya visto con los mecanismos de modo y grado.

2.4. El Estado Deseable : Los Intereses

Se ha planteado, al hablar de los niveles superiores de la arquitectura (Deliberativo y Social), la necesidad de permitir inhibir algunos comportamientos reactivos en determinadas circunstancias, en aras de alcanzar un objetivo o meta superior. En términos humanos es lo que conocemos como controlar una reacción. Sin embargo, esta necesidad no debe provocar que se pierdan las ventajas de una acción rápida ante determinadas situaciones críticas.

COGNITIVA propone un mecanismo para aunar estos objetivos introduciendo un nuevo elemento en la arquitectura, los *intereses*, que definen los límites entre los que el individuo desea que se encuentren los valores de sus estados transitorios, generalmente, emociones y estados físicos. Los intereses están relacionados, en un instante dado, con un y sólo un estado transitorio, y establecen además un límite superior y un límite inferior. Los intereses serán creados, modificados y eliminados por los niveles deliberativo y social, y el nivel reactivo sólo los consultará.

La capacidad de un individuo para controlar su comportamiento reactivo, esto es, los intereses, depende en gran medida de la personalidad del individuo. Un individuo *débil* no será tan capaz de soportar su *hambre* como un un individuo *fuerte.* Por tanto aparece una relación de influencia más, que se dará entre los rasgos de personalidad y los intereses. Esta influencia utilizará también los factores de modo y grado, y podrá definirse tanto hacia el umbral superior del interés como hacia el umbral inferior.

A modo de resumen, se muestran en la figura 2.4.1 las relaciones entre todos los componentes del modelo personal

2.5. Interpretación de las Percepciones del Agente

En la sección 2.1 se ha planteado la percepción del entorno como uno de los procesos más importantes del agente. La información recogida por los sensores llega al módulo cognitivo y allí es traducida a perceptos por el intérprete. Este intérprete es la interfaz entre el módulo cognitivo y los sensores. A partir de este punto, es preciso realizar dos tareas sobre los perceptos: por un lado, se ha de filtrar la información necesaria para cada uno de los niveles cognitivos de cara a los comportamientos reactivos, deliberativos y sociales; por otro lado, los perceptos tienen que actualizar las *creencias,* es decir, las estructuras de información del agente.

La primera función la realizan los propios procesos de percepción, que son los que deciden qué preceptos les interesan y buscan cambios solamente en los que tienen consecuencias sobre ellos.

Sin embargo, la segunda tarea puede ser bastante compleja. Podemos encontrarnos ante dos situaciones: (1) actualizar una creencia relativa al entorno, es decir, el objeto *n* pasa a estar en la posición *x*. Este tipo de actualizaciones no reviste complejidad; y (2) considerar la repercusión de esos sucesos sobre el estado emocional del agente, es decir, las creencias del modelo personal. Para este caso se van a considerar dos factores:

- **La expectación**, o cuánto espera el agente que un acontecimiento suceda. Por ejemplo, si ocurre un acontecimiento inesperado, el agente reaccionará con sorpresa.

- **El deseo**, o cómo es de beneficioso para el agente que tenga lugar el acontecimiento. Por ejemplo, si ocurre algo no deseado, la tristeza o incluso el miedo atraparán al agente.

La combinación de ambos factores constituye el concepto de *expectativa*, es decir, acontecimientos que el agente espera y desea con un cierto grado.

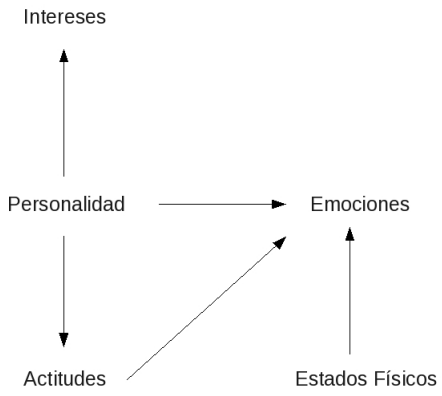

Figura 2.4.1: Influencia entre los elementos del Modelo Personal

2.6. Organización de las Acciones del Agente

De la misma manera que hemos profundizado en el intérprete, es el momento de analizar en detalle la interfaz entre COGNITIVA y los actuadores: el organizador. El organizador recoge las acciones propuestas por las distintas capas y, conservando su orden interno dentro de cada capa, las secuencia en una estructura que llamaremos *agenda.* La acción que ocupe la primera posición en un momento dado en la agenda será la elegida para ser enviada a los actuadores. En este momento hemos de hacer una breve explicación de lo que se define en COGNITIVA como acción. Como primera aproximación, una acción es un operador que se manda a los actuadores con unos parámetros determinados. Pero además se definen otros componentes como:

- Precondición, entendida como *lo que tiene que cumplirse* para poder ejecutar la acción en el momento de ser entregada a los actuadores.
- Operador, es decir la acción que queremos que sea realizada por los actuadores.
- Postcondición, o el resultado que el agente quiere que produzca la ejecución de la acción.
- Expectativas, que como ya hemos definido en la sección 2.5 son los acontecimientos que el agente espera (o no) y desea (o no) que sucedan como consecuencia de la acción.
- Caducidad, como el tiempo máximo que una acción puede tardar en ser ejecutada.

Además se dan en la arquitectura dos tipos de acciones:

- **Acciones Concretas**. Son acciones que los actuadores son capaces de interpretar y llevar a cabo sin necesidad de información o decisión adicional. Son, en este sentido, atómicas.
- **Acciones Abstractas**. Éstas son, sin embargo, acciones compuestas que los actuadores no son capaces de ejecutar directamente. Requieren de un proceso de descomposición en nuevas acciones que, de nuevo, pueden ser concretas o abstractas. Las acciones abstractas surgen cuando tenemos la necesidad de planificar una acción que no es posible concretar hasta el momento de ejecución. Estas acciones pueden aparecer tanto en el nivel reactivo como en los superiores.

Cada vez que el agente pase a ejecutar la siguiente acción más prioritaria, obtendrá las expectativas de la misma (tanto si es abstracta como concreta) y las incluirá en el conjunto de expectativas del agente, mantenido por el intérprete. Si la acción se ejecuta correctamente, el organizador pasa a buscar la siguiente; en cambio, si falla se desecharán las acciones relacionadas con la acción que falló.

2.7. Nivel Reactivo

El nivel reactivo dota al agente de la capacidad de dar una respuesta rápida a los cambios producidos en el entorno. Entendemos como cambios tanto los sucesos acaecidos que tengan influencia sobre el agente como el nuevo conocimiento adquirido relevante para el mismo. La principal característica del nivel reactivo es la velocidad de procesamiento y, por consiguiente, el breve tiempo de respuesta; sin embargo, en función del grado de involuntariedad de la acción distinguimos dos tipos de reacciones:

- **Procesado de Reflejos**. Este tipo de respuestas tienen una carga mínima de voluntariedad, son lo que comúnmente conocemos como *reflejos*. Un ejemplo sería agarrarnos a un asidero firme cuando nos sentimos caer. Estos reflejos vienen definidos por:
  - Disparadores, que se componen de las percepciones y valores de creencias del agente que desencadenan la reacción.
  - Justificadores, restricciones sobre los intereses que son necesarias para que la acción tenga sentido.
  - Acción Respuesta, conjunto de acciones que se generan cuando son ciertos los disparadores y justificadores.
- **Procesado de Reacciones Conscientes**. El agente también es capaz de elaborar *mini-planes* o secuencias de acciones de origen no deliberativo que, diseñadas a priori o aprendidas durante la ejecución, se han automatizado y que se generan para dar una respuesta rápida a situaciones que se dan con frecuencia. Un ejemplo sería la secuencia de acciones que hace una persona al levantarse. Esta secuencia, de tanto repetirse, no requiere de ninguna deliberación; el ser humano optimiza el tiempo al generarlas de forma reactiva.

Las acciones provenientes de estos procesos serán las más prioritarias de la arquitectura. En la figura 2.7.1 se muestra un esquema de la arquitectura reactiva completa.

2.8. Nivel Deliberativo

El objetivo de disponer de una capa deliberativa en la arquitectura es el de dotar al agente de la capacidad de generar acciones de una forma racional en situaciones que no son demasiado críticas desde el punto de vista temporal. Estas acciones se enmarcan dentro de planes, diseñados para lograr un objetivo a largo plazo.

En este nivel consideraremos la generación de planes en función únicamente de las capacidades propias del agente, en la siguiente sección trataremos los planes que tienen en cuenta la componente social, es decir, que solicitan servicios a otros individuos.

2.8.1. Mantenimiento de los Objetivos del Agente : Metas

Las metas son los objetivos que el agente establece como necesarios a medio y largo plazo. Están caracterizadas por los siguientes componentes:

- **Situación Objetivo**. Es una expresión lógica que expresa el estado al que el agente quiere llegar.
- **Importancia**. Cuantifica la prioridad de la meta.
- **Tiempo de generación**. Instante en que se generó la meta, sirve para determinar si una meta sigue o no vigente.
- **Caducidad**. Tiempo máximo en el que una meta deberá ser satisfecha.

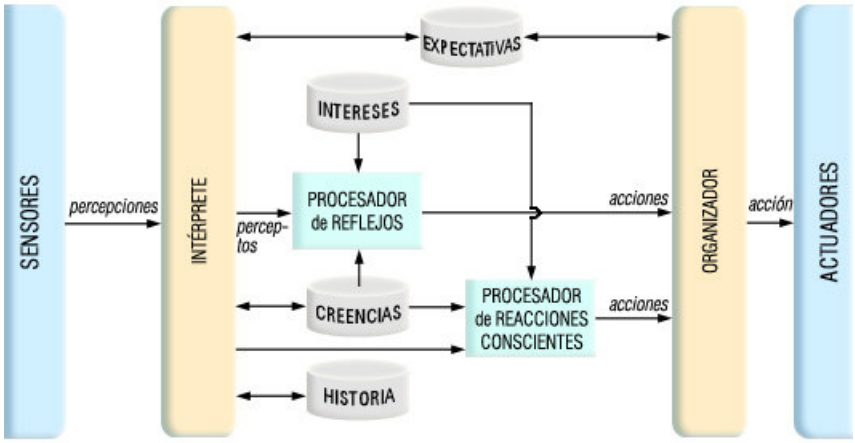

Figura 2.7.1: Arquitectura Nivel Reactivo

- **Estado actual**. Estado en el que se encuentra la meta. Puede ser:

  1. En espera, La meta ha sido generada y está en espera de ser planificada.
  2. En procesamiento, el planificador está intentando elaborar un plan para satisfacer la meta.
  3. Inalcanzable, el planificador es incapaz de elaborar un plan que satisfaga esa meta.
  4. Planificada, el planificador ha elaborado un plan que satisface esa meta.
  5. En ejecución, el organizador ha recogido el plan elaborado por el planificador. Mientras siga habiendo acciones del plan en el organizador, la meta seguirá en ejecución.
  6. Cancelada, la meta ha dejado de ser interesante para el agente.
  7. Alcanzada, se ha ejecutado con éxito la última acción de un plan trazado para alcanzar la meta.

En la figura 2.8.1 se muestra el ciclo de vida de las metas de un agente.

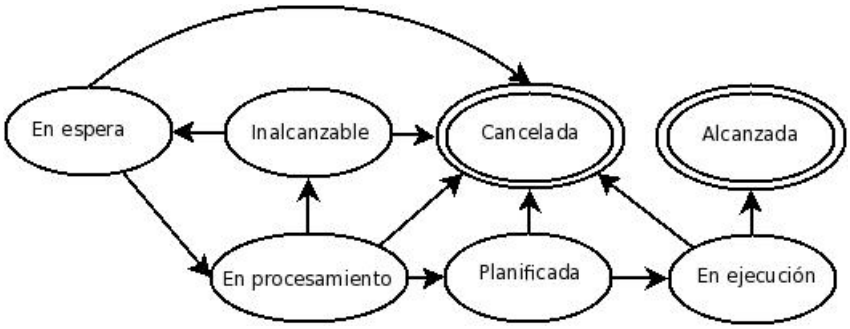

Figura 2.8.1: Ciclo de vida de una meta

2.8.2. Generación de Metas Resolubles por el Propio Individuo

Las metas en esta capa son generadas por un componente llamado *generador de metas deliberativo*. El generador puede establecer metas desde dos fuentes distintas: De un origen externo, debido a cambios en el entorno; o desde una fuente interna, con motivo de variaciones en el modelo personal. La mayoría de las veces el origen será una mezcla de ambos.

Al generar una meta, se comprueba si ya existe en el conjunto de metas alguna con el mismo objetivo final. Si existe, y la importancia es la misma, se puede optar por actualizar el tiempo de generación, modificar la caducidad o dejarla como está; en cambio, si la importancia es mayor, se actualiza el valor de la importancia de la meta al nuevo valor. Por otro lado, si no hay ninguna con el mismo objetivo, se inserta la nueva meta en el repositorio.

Además, el generador de metas se encarga del mantenimiento de las metas, comprobando cíclicamente la vigencia de las mismas y si han sido o no alcanzadas. Además, si en algún momento una meta deja de ser interesante, el generador la pondrá en el estado cancelada (aunque siguiera vigente). Por último controla las metas inalcanzables, decidiendo si las condiciones han cambiado lo suficiente para intentar una replanificación.

2.8.3. Diseño de Planes desde la Perspectiva Deliberativa

Las metas, independientemente de quién las haya generado, pueden ser resueltas desde una perspectiva autónoma del agente, o bien desde una perspectiva social. En este apartado describiremos la planificación contando tan sólo con las capacidades propias del agente.

El encargado de esta tarea es el *planificador deliberativo.* El planificador recibe una meta en estado de espera y va tratando de elaborar un plan proponiendo acciones que conduzcan al estado objetivo. En esta tarea le ayudará su homólogo, el *planificador social,* debido a que los planes se construyen intercalando acciones de ambos. Si se consigue elaborar uno o más planes, la meta pasa a estar planificada. Más adelante, cuando el plan se incluya en la agenda del organizador, la meta pasará a estar en ejecución, y así seguirá hasta que se alcance o bien hasta que sea cancelada.

El planificador deliberativo se encarga también de controlar los umbrales de los intereses, de forma que el nivel reactivo no interfiera en sus objetivos; ya se ha comentado que para alcanzar un fin superior puede ser necesario inhibir algunas reacciones, como por ejemplo somos capaces de controlar los nervios en una entrevista de trabajo de cara a conseguir el puesto.

Sin embargo, el planificador no podrá directamente ejecutar este control debido a que sólo ha de hacerse efectivo cuando el plan pase a ejecución; por tanto, su tarea será ordenar la adecuación de los intereses al organizador. La forma de hacerlo es intercalando en el plan enviado al organizador acciones que modifiquen y restauren los umbrales de los intereses durante el periodo crítico en el que se puedan disparar reacciones.

Las acciones planificadas pueden ser también de tipo abstracto, como ya vimos en la sección 2.6, y por tanto se requiere de un componente auxiliar que se denomina *razonador deliberativo* que, a partir de las creencias e intereses del agente, descomponga estas acciones. En el nivel reactivo esta tarea era realizada por el organizador, sin embargo, ahora hemos de tratarlas en un componente aparte debido a que no tiene porque cancelarse el plan ante un resultado inesperado; el razonador trataría de buscar alternativas que proporcionaran otra forma de descomponer la acción.

En la figura 2.8.2 podemos observar la morfología de la arquitectura deliberativa al completo, similar a la reactiva, añadiendo los componentes propios.

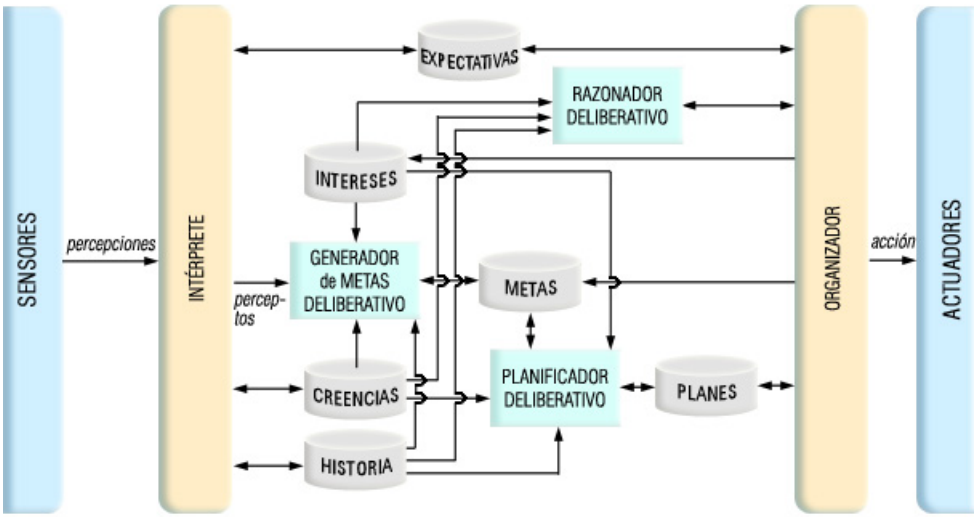

Figura 2.8.2: Arquitectura Nivel Deliberativo

2.9. Nivel Social

2.9.1. Consideración de la Existencia de Otros

El tercer y último nivel de la arquitectura COGNITIVA es el que dota de capacidades sociales al agente. Sus fundamentos de partida son similares a los del nivel deliberativo, establecer metas que guíen la conducta del agente y buscar planes que alcancen dichas metas. La diferencia es que el nivel social trata de aprovechar la existencia de otros individuos empleando sus capacidades para la consecución de las metas.

La decisión de situar en un nivel jerárquico más alto al nivel social obedece al criterio de *indispensabilidad*, es decir, cuanto más bajo es el nivel, más imprescindible es para la supervivencia del agente. Un agente no necesita un nivel social en un entorno donde no existan individuos, o donde no interactúe con ellos. Otra manera de plantear la superioridad jerárquica del nivel social sobre el deliberativo es entendiendo al nivel deliberativo como un caso particular del social donde sólo existe un individuo, el propio agente. Por lo demás, el nivel social hace uso de las mismas estructuras de información que el resto de niveles, y su interacción con el intérprete y el organizador será similar.

2.9.2. Generación de Metas a Partir de la Interacción con Otros Individuos

El nivel social participa junto con el deliberativo en la generación y mantenimiento de las metas del conjunto de metas. Esto es así porque en la arquitectura no se distingue entre metas *sociales* y metas *deliberativas*. Las metas generadas desde el punto de vista social tratarán de satisfacer necesidades del agente en su interacción con el resto de individuos, pero en el resto de sus características serán iguales que las metas generadas desde el deliberativo. Para generarlas se dispondrá, análogamente al nivel deliberativo, de un *generador de metas social;* cuyas funciones serán las mismas que el deliberativo.

2.9.3. Diseño de Planes desde la Perspectiva Social

Como ya se mencionó en la sección 2.8.3, al hablar del diseño de planes desde el deliberativo, el *planificador social* y el deliberativo, suman sus esfuerzos a la hora de elaborar los planes, proponiendo acciones que se van intercalando, a la hora de satisfacer una meta. Los planes son, por tanto, generados cooperativamente y se almacenan en un conjunto de planes común. El resto de funciones y responsabilidades del planificador social son iguales a las del deliberativo.

Así mismo, el nivel social posee un componente para descomponer las acciones abstractas cuando sea necesario, el *razonador social.* Este razonador descompone acciones abstractas que requieren la interacción con otros individuos. Por ejemplo, una acción abstracta sería *solicitar* y en el momento en que llegase al organizador, el razonador la descompondría en *enviar_solicitud* , que será una acción concreta, y en *tratar_respuesta* que a su vez será una abstracta que analizaría la respuesta a la solicitud. Si se entrara en un proce-

so de negociación, el número de acciones en que llega a descomponer la acción original *solicitar* es totalmente indeterminado.

Finalmente en la figura 2.9.1 podemos ver la arquitectura del nivel social al completo, totalmente análoga a la arquitectura deliberativa.

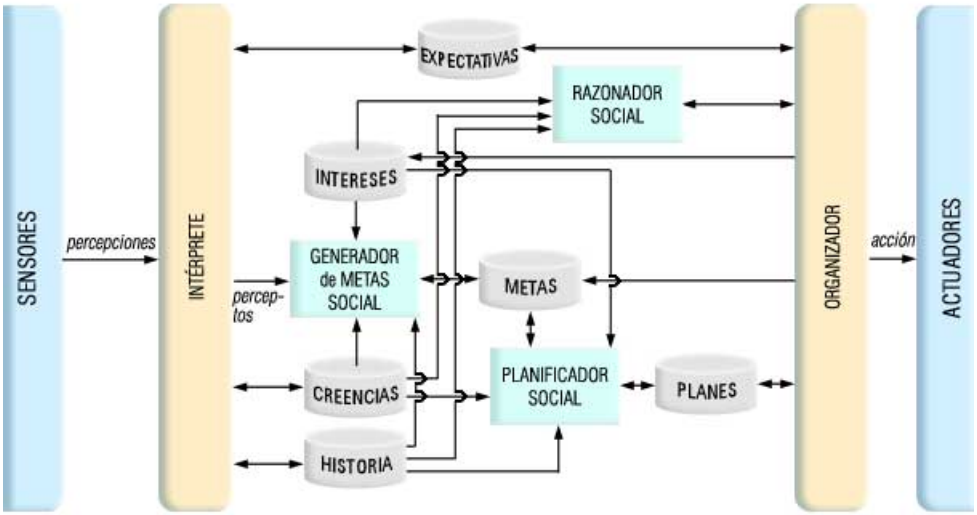

Figura 2.9.1: Arquitectura Nivel Social

2.10. Análisis de la Arquitectura

Para abordar el objetivo del proyecto, esto es, dotar a la arquitectura COGNITIVA de los componentes necesarios para satisfacer las necesidades actuales, hemos de realizar un análisis previo, tratando los puntos fuertes y las carencias de la arquitectura. Este análisis se recoge de forma esquemática en el cuadro 2.1.

2.10.1. Velocidad de Respuesta

Uno de los parámetros fundamentales para evaluar la calidad del software es la velocidad de respuesta. COGNITIVA proporciona una capacidad de reacción muy alta frente a sucesos en el entorno gracias a su nivel reactivo. Sin embargo, esta velocidad se pierde en la implementación debido a las estructuras de datos elegidas para manejar la información. Las creencias, relaciones entre creencias, intereses y reglas del agente están estructuradas

en listas enlazadas, que debido a la alta frecuencia con la que se recorren, ralentizan mucho el sistema. La propuesta que hacemos, recogida en [Cabeza, S.N.], es implantar una ontología que maneje el conocimiento del agente.

2.10.2. Nivel Social

El nivel social, pese a estar planteado, no ha sido desarrollado en COGNITIVA ya que los trabajos de [Leal, 2005, Molina, 2005] abarcaron tan sólo el nivel reactivo y deliberativo. Pese a todo, COGNITIVA tiene en cuenta la existencia de otros individuos en el sentido en que percibe a otros individuos y actúa en función a información acerca de los mismos. El objetivo del presente trabajo es desarrollar la interacción social entre los agentes para que trabajen de forma cooperativa, sentando las bases del nivel social.

2.10.3. Creencias

La gran variedad de creencias manejadas por el agente hace que sea necesario el manejo de tipos de datos muy heterogéneos. En la arquitectura se manejan tan sólo creencias de tipo numérico. Sin embargo se dispone de un tipo genérico *valor* que debe permitir la instanciación de nuevos tipos de datos. En este trabajo se tratará de ampliar esta capacidad de cara a abordar, entre otros asuntos, las creencias sobre la situación actual.

2.10.4. Expectativas

Uno de los mecanismos de COGNITIVA que más credibilidad en el comportamiento proporciona al agente es el de las expectativas. Sin embargo, creemos que ha sido abordado sólo superficialmente, y no se han tenido en cuenta determinados controles y consideraciones sobre las expectativas que garanticen, entre otras cosas, que su influencia sobre las emociones del agente no sea excesiva.

2.10.5. Concreción

La especificación descrita en este capítulo abarca tan sólo el nivel más general de la arquitectura cognitiva, ya que se indican solamente las características básicas comunes a cualquier agente COGNITIVA. El resto de aspectos deben ser descritos en niveles de refinamiento posteriores, conocidos como concreciones. Esta división proporciona ciertas ventajas:

- La arquitectura propone las estructuras de información y los datos, pero los formatos se especifican posteriormente.
- El modelo personal del agente se describe en la arquitectura, pero no se detalla, por lo que se puede trasladar a cualquier escenario.
- El resto de componentes tan sólo se especifican funcionalmente, y en referencia a los interfaces entre los mismos y con las estructuras de datos. Su especificación detallada depende de cada aplicación.
- Los niveles de la arquitectura (reactivo, deliberativo y social) pueden simplificarse, o incluso ampliarse.
- Los valores con los que va a funcionar la arquitectura son dependientes de cada aplicación específica.

Las etapas que trasladan el modelo teórico a la implementación específica son dos:

1. Concreción Funcional. En esta etapa se especifican detalladamente todas las funciones *primitivas* que se realizan en el módulo cognitivo, necesarias para que dicho módulo funcione, pero totalmente independientes del contexto. Además se especifica el formato de datos a emplear en las estructuras de información del agente.
2. Concreción Contextual. Esta etapa, a realizar después de la Concreción Funcional, consiste en la especificación de los valores concretos de cada uno de los elementos de la arquitectura, de acuerdo con el contexto de aplicación del agente y el problema concreto a resolver.

La división en concreciones (funcional y contextual) del desarrollo de entornos basados en COGNITIVA ha sido un factor clave para optimizar el tiempo de desarrollo de los mismos. Sin embargo, la implementación no respeta la división estricta entre ambas concreciones, produciéndose acoplamientos en diferentes partes del código que no permiten independizar la Concreción Funcional del contexto. Desacoplar totalmente ambas partes es un objetivo prioritario del presente proyecto.

2.10.6. Ficheros de Inicialización

Gracias a la interfaz basada en ficheros de configuración de texto, cualquier desarrollador puede elaborar su propia Concreción Contextual, simplemente rellenando las plantillas

con la incialización de las creencias, reglas y demás estructuras de información del agente. Sin embargo, el formato establecido para estos ficheros es excesivamente cercano a la máquina, siendo en ocasiones necesario conocer aspectos internos del código para poder rellenarlos. *Humanizar* estos formatos será un objetivo más del trabajo.

2.10.7. Planificador

La elaboración de planes en el nivel deliberativo se desarrolló utilizando un planificador jerárquico HTN que posibilita la consideración de diferentes alternativas para lograr una meta. Aunque cubre las necesidades de la arquitectura, el mantenimiento y ampliación con nuevos planes es una tarea compleja debido a ser un planificador SHOP2 con planes en lenguaje LISP, que requiere conocer a fondo su arquitectura y el lenguaje para cualquier cambio. Implantar un planificador con una API más manejable es una de las extensiones fundamentales para COGNITIVA.

2.10.8. Código

En cualquier desarrollo software, uno de los equilibrios fundamentales es el establecido entre el tiempo de desarrollo del código y la capacidad de mantener el mismo a posterioridad. En la primera versión de COGNITIVA se decidió optar por priorizar la primera variable debido a unos plazos de entrega muy estrictos. Es un objetivo transversal a todo nuestro trabajo el garantizar un código legible, mantenible y ampliable de cara a nuevos desarrolladores que puedan aportar más extensiones.

Análisis de la Arquitectura	Pros	Contras
Velocidad de Respuesta	Nivel Reactivo	Listas
Nivel Social	Percepción Individuos	Interacción Social
Creencias	Clase Valor	Situación Actual
Expectativas	Realismo	Influencia Excesiva
Concreción	Funcional y Contextual	Acoplamiento
Ficheros Inicialización	Texto	Formato
Planificador	HTN	SHOP2 (LISP)
Código	Desarrollo rápido	Mantenimiento

Tabla 2.1: Análisis de la Arquitectura

Capítulo 3

REDISEÑO Y EXTENSIÓN DE LA ARQUITECTURA

3.1. Refactoring

Tras analizar en detalle los componentes de la arquitectura definidos en el capítulo 2 la tarea más compleja fue comprender e identificar los mecanismos allí planteados y su traslado a la implementación. En el transcurso de este periodo, se detectaron gran cantidad de problemas en relación con la legibilidad, el mantenimiento y la capacidad de extensión del código desarrollado.

Empezando por una ausencia casi completa de documentación, se estableció la necesidad prioritaria de mejorar la calidad del código antes de la implantación de cualquier nueva capacidad.

Ahora bien, este proceso había de desarrollarse garantizando la conservación de todas las funcionalidades y, más allá, exigiendo los mismos resultados en las pruebas. Para asegurar este compromiso se establecieron unos escenarios de prueba que se desarrollan en profundidad en [Cabeza, S.N.].

COGNITIVA está desarrollado en JADE, una aproximación a lo que deberán ser en un futuro los lenguajes orientados a agentes. De momento, JADE es un framework para el desarrollo de agentes que emplea el lenguaje orientado a objetos Java, con todo lo que ello conlleva. Es por esto que la arquitectura tiene un diseño orientado a objetos, incorporando los componentes propios de agentes que proporciona JADE.

En esta sección trataremos los diferentes problemas encontrados a la hora de analizar la

arquitectura basándonos en los fundamentos de la orientación a objetos.

3.1.1. Uso de herencia, reparto de responsabilidades y clases innecesarias

3.1.1.1. Herencia

La herencia es el mecanismo mediante el cual una clase, al declararse hija de otra, hereda sus métodos y atributos con el objeto de crear estructuras jerárquicas que ahorren esfuerzo en el desarrollo. Además permite adaptar dichos métodos (sobreescribir) y añadir otros nuevos, específicos de la clase hija.

En la primera versión de COGNITIVA no se utilizaba apenas la herencia, aunque había una gran variedad de estructuras de clases en las que se podía aplicar.

Uno de los casos más paradigmáticos es el de las *listas,* que, como ya se ha señalado, eran las estructuras de datos que almacenaban la información del agente. El agente disponía de listas de *creencias, reglas, acciones, expectativas,...* y la manera en la que los desarrolladores las implementaron fue creando clases diferentes para cada tipo de lista. Estas clases, tenían un único *atributo* que era de la clase ListaEnlazada (genérica) y sus métodos no realizaban otra tarea más que llamar a los métodos genéricos de dicha clase.

Esta situación de duplicidad del código se daba en las doce clases que, conceptualmente, eran listas de objetos. La primera solución que se planteó para solucionar este problema fue directamente eliminar estas clases y trabajar únicamente con la clase ListaEnlazada. Esta es la solución que se adopta en las clases que no añaden atributos y/o métodos a la clase original ListaEnlazada, las clases eliminadas Precondición y Postcondición. Sin embargo, las clases en las que había métodos propios, se declaran como hijas de ListaEnlazada, y por tanto, su contenido corresponde únicamente a los métodos que añaden. En la figura 3.1.1 se muestra la jerarquía resultante de aplicar herencia en las clases de *listas* de la arquitectura.

La otra clase de la arquitectura en la que la herencia debía ser no sólo por eficiencia, sino por necesidad, aplicada es la clase AgenteSimple. La clase AgenteSimple recoge todos y sólo esos componentes de la Concreción Funcional que son necesarios en un agente COGNITIVA. Es decir, la clase AgenteSimple es el esqueleto sobre el que deben construirse los agentes de cada Concreción Contextual añadiendo sus particularidades.

Sin embargo, en la primera versión, si bien esta clase existía, sus atributos, métodos y comportamientos estaban copiados en cada uno de los agentes contextuales de forma que

se producía el temido acoplamiento entre Concreción Funcional y contextual. De esta manera, una de las primeras tareas ha sido declarar a cada agente contextual como hijo del AgenteSimple, heredando sus comportamientos y sobreescribiendo su método *setup* de inicialización, así como añadiendo sus comportamientos particulares.

Además, en el mismo paquete en que está la clase AgenteSimple, el paquete Agente Básico, se da un caso similar con la clase Actuador. El desarrollo de la clase Actuador se explica en la sección 3.1.3, pero el resultado, que es una clase Actuador padre y una clase Actuador hija por cada agente contextual, aplica el mismo principio de desacoplar la parte contextual de la funcional.

La estructura final del paquete Agente Básico, que incluye además los Lectores, el Organizador, el Actuador y las clases Modifica (Creencia e Interés) se muestra en la figura 3.1.1.1 que expone la jerarquía incluyendo ejemplos de clases contextuales. Se ha representado a los comportamientos como métodos en las clases Agente adaptando la notación UML.

A modo de resumen, se enumeran las estructuras jerárquicas de herencia presentes en la arquitectura, señalando las que se han incorporado en esta versión:

- Jerarquía de la clase AgenteSimple (Nueva).

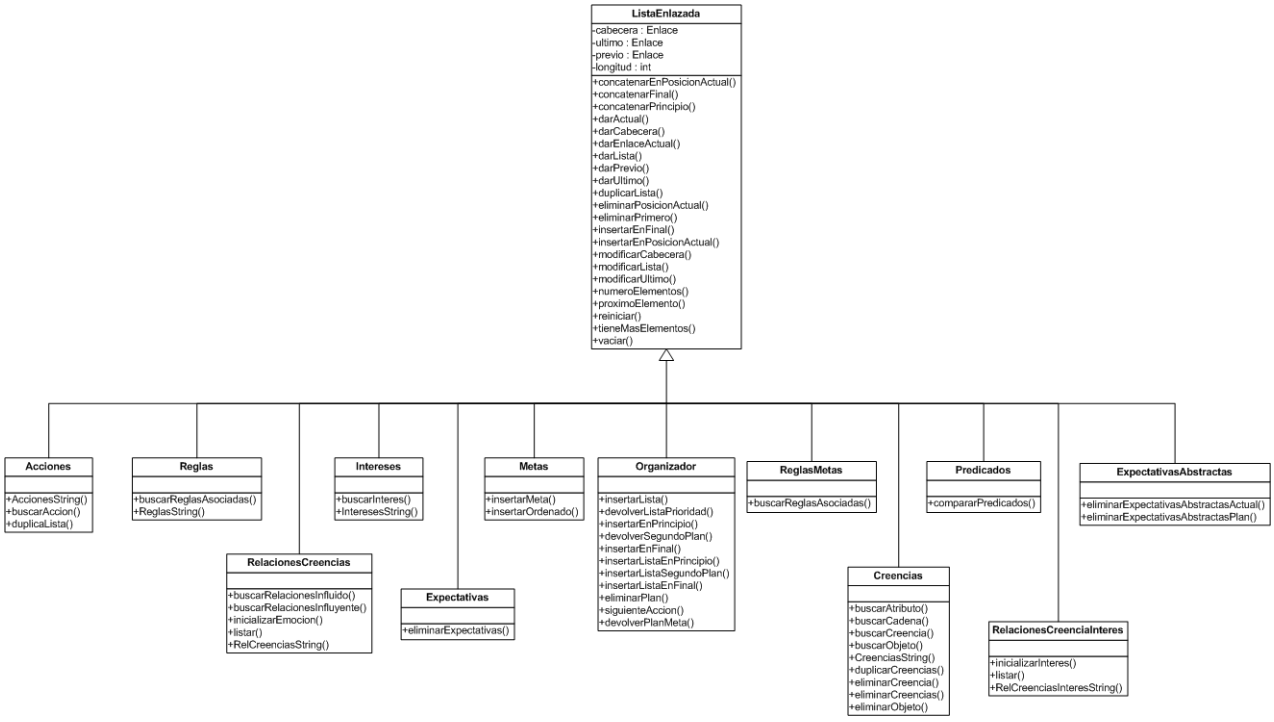

Figura 3.1.1: Jerarquía ListaEnlazada

- Jerarquía de la clase Acción. Ver en detalle en la sección 3.1.2 (Nueva).
- Jerarquía de la clase Actuador (Nueva).
- Jerarquía de la clase Creencia.
- Jerarquía de la clase ExpLogica.
- Jerarquía de la clase ListaEnlazada (Nueva).
- Jerarquía de la clase Valor. Ver en detalle en la sección 3.1.4.

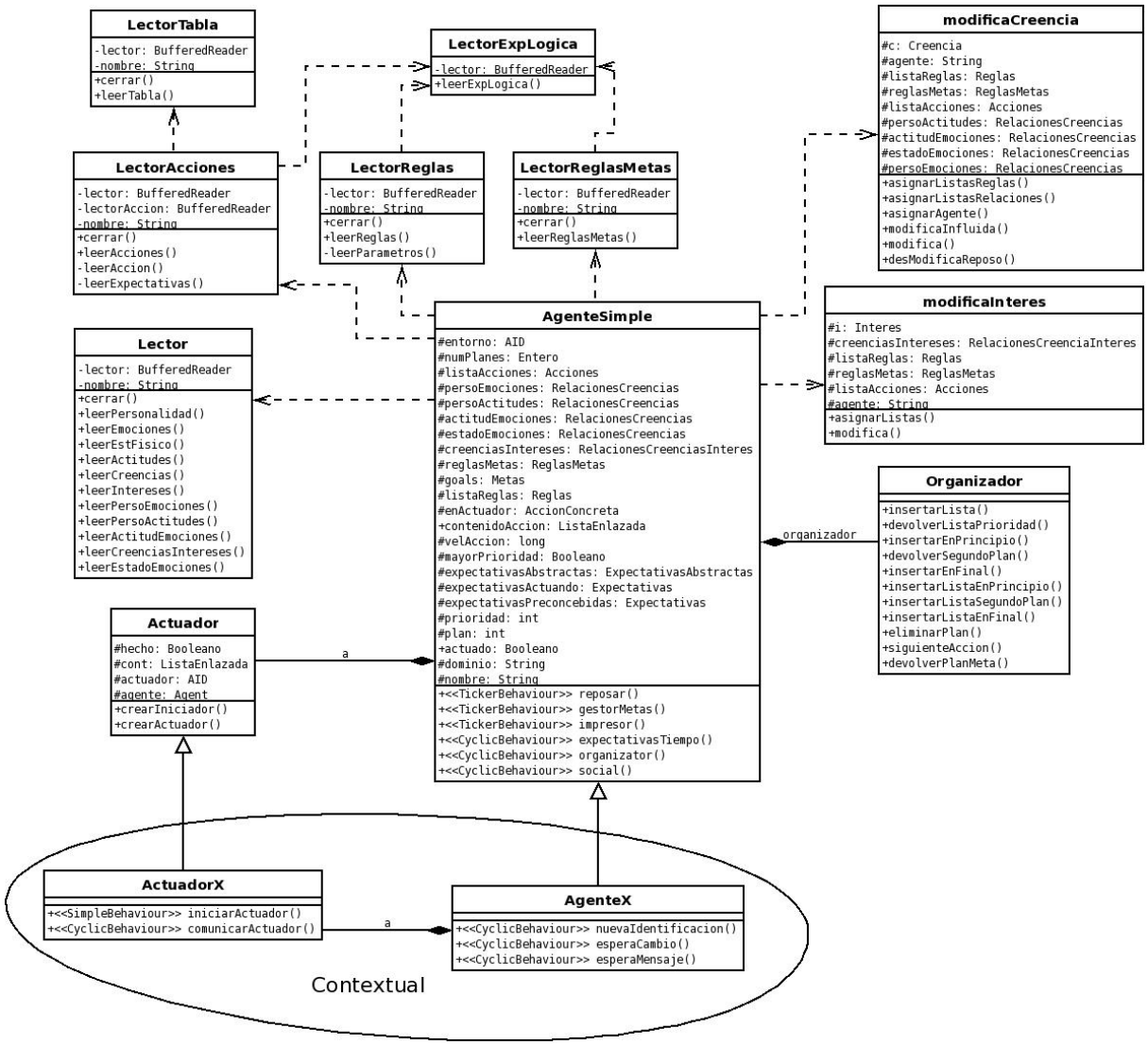

Figura 3.1.2: Diagrama Clases Agente Básico

3.1.1.2. Responsabilidades

Para conseguir un diseño orientado a objetos correcto, una de las tareas principales es realizar un reparto de responsabilidades adecuado, es decir, que cada clase se encargue de todo y sólo lo que le concierne.

En la primera versión de COGNITIVA se detectó un reparto de responsabilidades poco adecuado; uno de los principales defectos era crear clases para lo que sólo eran métodos. Concretamente este error se cometía en dos clases:

- **Razonadores**, los razonadores eran objetos asociados a un objeto de la clase AccionAbstracta y que sólo tenían un método, *razonar*. El agente disponía de un objeto de la clase RazonadorReactivo que era una colección de objetos de la clase RazonadorConcreto, y que a su vez era la clase padre de la que heredaban los razonadores asociados a cada acción. Toda esta amalgama de objetos-método se eliminó al crear una clase por cada tipo de acción (ver sección 3.1.2) e incorporar el método *razonar* a la clase AccionAbstracta; de esta manera, cada clase contextual sobreescribía el método en función de la descomposición de la acción.

- **Evaluadores**, al igual que los razonadores, los evaluadores (EvaluadorReglas y EvaluadorReglasMetas) eran clases con un único método que era, por supuesto, *evaluar*. Aplicando el mismo criterio que con AccionAbstracta, se incorporó el método evaluar a la clase Regla y ReglaMeta respectivamente, eliminando los evaluadores.

Otro de los puntos en los que el reparto de responsabilidades no era adecuado era en los lectores y ficheros de configuración puesto que se mezclaban acciones, reglas y reglas metas en un mismo fichero con un mismo lector. Separarlo en tres ficheros y lectores, uno para cada una de las estructuras, fue una forma de garantizar que las clases tuvieran un adecuado balance de responsabilidad.

3.1.1.3. Clases innecesarias

Finalmente, fue una sorpresa observar que había una gran cantidad de clases que resultaban ser superfluas, algunas por los motivos expuestos en los apartados anteriores y otras simplemente porque no se utilizaban. Posiblemente estas últimas formaban parte de los cimientos de futuros componentes del proyecto que nunca se llegaron a implementar.

Además debido a la sustitución del planificador existente por el nuevo, todas las clases que componían el antiguo fueron eliminadas.

Seguidamente se listan todas las clases eliminadas de la versión de COGNITIVA original, junto con los motivos que llevaron a desecharlas.

- Las clases EvaluadorReglas y EvaluadorReglasMetas (ver apartado anterior).
- Las clases RazonadorReactivo, ElementoRazonador, RazonadorConcreto, RazonadorConcreto*Accion* (ver apartado anterior).
- Las clases ConsecuenteExpectativa y ElementoConsecuenteExpectativa. No se utilizaban, junto con las clases del punto anterior formaban el paquete Reaccion que fue así eliminado.
- El paquete Deliberativo fue eliminado trasladando sus clases (Meta, ReglaMeta,..) a EstructurasReglas al ser estructuras análogas a las allí presentes.
- Las clases Atributo y Objeto fueron eliminadas del paquete EstructurasCreencias al estar ya duplicadas en EstructurasDatos. Lo mismo ocurrió con la clase Actuador en el paquete Conversores, quedando sólo la versión del paquete Agente Básico.
- Las clases Precondición y Postcondición fueron eliminadas al ser redeclaraciones de ListaEnlazada sin métodos propios.
- Las clases Ontología y UtilidadesAgente al no ser utilizadas. El paquete Ontología se mantuvo para albergar las API's de la ontología (ver en la sección 3.8).
- El paquete Planificador y todas sus clases fueron eliminadas al incorporar el nuevo planificador, cuya API se localiza en el paquete envoltorioplanificador.

3.1.2. Una clase por Acción y Jerarquía de Acciones (Nivel Reflexivo)

Continuando con la línea de de lo expresado en la sección 2.10 dentro del apartado de concreción, tras el análisis se señaló a la clase Acción como uno de los puntos de mayor acoplamiento entre las concreciones.

3.1.2.1. Clase y paquete por cada Acción

En la primera versión de COGNITIVA, las acciones que podía realizar el agente eran objetos instancia de la clase AccionAbstracta o de AccionConcreta (ambas hijas de la clase Acción) en función del tipo de acción que se definió en la sección 2.6. Por tanto el código de los objetos era el mismo, independientemente de la acción que representaran.

Sin embargo, las acciones concretas definen dos métodos cuyo código es totalmente dependiente de contexto, esto es, de la acción que realizan. Estos métodos son *parametrosDinamicos* y *actuadorInterno* que sirven, respectivamente, para corregir dinámicamente los parámetros de las acciones (ver sección 3.7.2) en función del modelo personal y para llevar a cabo acciones de carácter reflexivo (ver siguiente apartado). La forma de resolver está situación era incluyendo instrucciones if-then-else que discriminen sobre el nombre del acción, con el código de la acción contextual en cada rama, produciendo un acoplamiento total entre ambas concreciones.

Por otro lado, durante la realización de pruebas de la arquitectura para garantizar que los cambios no modificaban los resultados, apreciamos que se perdía una gran riqueza de expresión en las emociones del agente al tener una única tabla de expectativas (ver en la sección 3.2) para cada agente. Por ejemplo, no debía ser lo mismo para la *alegría* del agente que una expectativa *deseada y esperada* se cumpliera para la acción *huir* que para la acción *mover;* es decir, el grado y el modo de influencia de las expectativas sobre las emociones del agente no debía ser el mismo para todas las acciones.

De esta forma, debido a la necesidad de desacoplar en la clase Acción lo funcional de lo contextual junto con la necesidad de tener una tabla de expectativas por cada acción, se decidió crear una clase por cada acción. Estas clases son hijas de la clase padre que defina su tipo (ver siguiente apartado), y sobreescriben los métodos con su particularidades contextuales. Además están incluidas en un paquete con el nombre de la acción que tiene todos los ficheros de configuración de la acción (ver en la sección 3.2).

3.1.2.2. Jerarquía de Acciones. Nivel Reflexivo

En [Imbert, 2005] ya se plantea la posible ampliación de la arquitectura con nuevos niveles, planteando el caso de un *nivel reflexivo* que maneje acciones que provoquen actualizaciones de las creencias del propio agente debido a reflexiones sobre su propia actuaciónn, sobre las alternativas de actuaciónn que se le ofrecen, sobre los efectos de su actuaciónn...

Sin pretender desarrollar dicho nivel, ha surgido la necesidad de incorporar a la arquitectura un tipo de acciones que tengan consecuencias únicamente sobre el modelo personal del agente y se traten de forma interna. Es lo que se ha dado en llamar *acciones internas*. El primer ejemplo de este tipo de acciones fueron las acciones establecidas por los niveles superiores de la arquitectura para modificar los umbrales de interés de las emociones en los periodos críticos en que se pueden disparar reacciones. Pero, además, son de este tipo las acciones necesarias para modificar la situación actual cuando cambia el estado interno del agente (ver secciones 3.5 y 3.7.3).

Estas acciones no siguen el ciclo habitual de las acciones concretas, esto es, pasar de la agenda del organizador a los actuadores, y de ahí aplicar sus efectos sobre el entorno. Las acciones internas al ser seleccionadas por el organizador no van a los actuadores sino que ejecutan el método actuadorInterno, mencionado en el apartado anterior, y terminan. Una característica muy importante de las acciones internas es que aunque se cancele un plan, eliminando del organizador todas las acciones de dicho plan, las acciones internas no deben eliminarse porque podrían dejar inconsistente el estado del agente. Por ejemplo, si se ejecuta *empiezaHuir* y luego se cancela el plan, el agente permanecería en estado *huyendo* aunque ya no tuviera sentido; es preciso ejecutar la correspondiente *terminaHuir* para devolver el estado interno a su valor inicial.

En consecuencia a estos planteamientos, se redefinió la jerarquía de la clase Acción incorporando dos clases hijas de la clase AccionConcreta:

- Acción Externa, para las acciones que van a los actuadores y tienen consecuencias sobre el exterior.
- Acción Interna, para las acciones *reflexivas.*

En la figura 3.1.3 se puede ver la nueva jerarquía de tipos de acciones incluyendo los posibles ejemplos de acciones contextuales.

3.1.3. Optimización de métodos y comportamientos. Actuador

En esta subsección se recogen las optimizaciones en el código que, por su carácter heterogéneo, no pueden clasificarse dentro de otro apartado de la sección. En el primer apartado se detallarán las mejoras realizadas en varios de los métodos más relevantes y la estructuración de algunas partes del código. También se especificarán las interacciones entre los agentes y el entorno a través de los protocolos establecidos entre ellos.

En el segundo apartado se explicarán en profundidad los cambios realizados en el Actuador y se entrará a fondo en los comportamientos del agente a través de los diagramas de comportamiento.

3.1.3.1. Optimización

Además del refactoring realizado en la sección 3.1.1, se ha aplicado el principio de programación orientada a objetos que señala que en cada fichero de código debe implementarse una y sólo una clase. Siguiendo esta idea, las clases ModificaCreencia y ModificaInteres que estaban declaradas en el fichero AgenteSimple.java, donde se declaraba la clase AgenteSimple, han sido trasladadas a ficheros independientes: modificaCreencia.java y modificaInteres.java.

Esta factorización ha implicado que los atributos que compartían estas clases con AgenteSimple, hayan tenido que ser pasados como parámetro, aumentando el tamaño del código de las llamadas a estas clases. Aún así, el cambio garantiza un mejor mantenimiento, así como permite aislar cualquier error en los algoritmos de modificación de creencias e intereses dentro de sus respectivas clases.

También se ha optado por hacer un rediseño en algunos de los comportamientos de cada agente y de los protocolos entre ellos. Se han utilizado modelos pertenecientes al Diseño

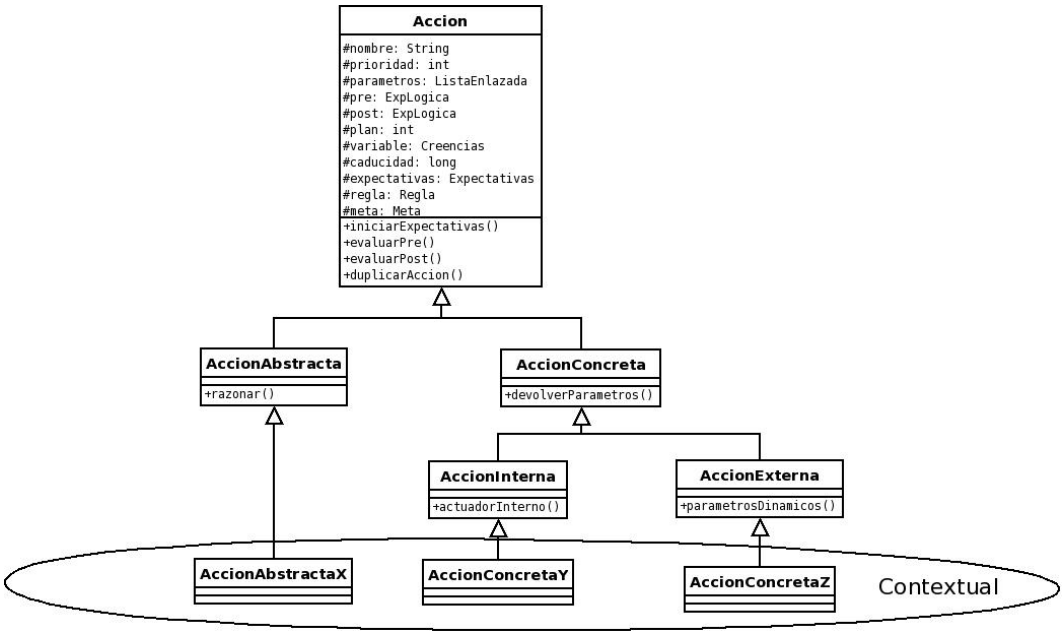

Figura 3.1.3: Jerarquía Acción

Detallado Extendido visto en clase que cubre el hueco existente entre la propuesta de Diseño Detallado de Gaia y la programación, de acuerdo con las restricciones del entorno de implementación (JADE). Estos modelos son los Diagramas de Comportamiento y los Protocolos Detallados.

Los Diagramas de Comportamiento permiten definir formalmente con una notación tabular los comportamientos (*behaviour en JADE*) de cada agente indicando:

- Nombre del Comportamiento: nombre para identificar este comportamiento.
- Agente: agente al que está asociado el comportamiento.
- Protocolo: protocolo asociado al comportamiento.
- action: descripción de lo que se espera que haga el comportamiento.
- onStart: descripción de lo que se espera que haga el comportamiento antes de ejecutar el método action.
- onEnd: descripción de lo que se espera que haga el comportamiento una vez que haya finalizado.
- done: descripción de la condición de finalización del comportamiento.

Entre los comportamientos modificados encontramos el caso de los comportamientos mantenerCreencias y organizator; el primero se encargaba de actualizar las creencias modificadas como resultado de una acción interna, y el segundo de buscar la siguiente acción en el organizador y descomponerla, si era abstracta, o mandarla al actuador, si era concreta y externa. Se decidió fusionar ambos comportamientos para garantizar que se ejecuten de forma secuencial, y para agrupar en un mismo comportamiento el tratamiento de todos los tipos de acciones: abstractas, internas y externas.

En el cuadro 3.1 se encuentra el diagrama del comportamiento organizator resultado de la fusión.

Por otro lado, los Protocolos Detallados son diagramas de interacción entre agentes basados en una extensión de los *Agent Interaction Protocols* de AUML (ver [Odell et al., 2001]) cuyo objetivo es especificar el intercambio exacto de mensajes correspondiente a cada protocolo identificado.

A través de estos diagramas, se rediseño el protocolo NaceAgente, en el que se realizaban algunos mensajes irrelevantes que complicaban la comprensión del protocolo. En la figura

3.1.4 se muestra el diagrama de protocolo detallado con el intercambio de mensajes de la versión final y en la figura 3.1.5 el del protocolo CambioAgente, que especifica como un agente ejecuta sobre el Entorno una acción, y éste a su vez comunica el resultado al agente y al resto.

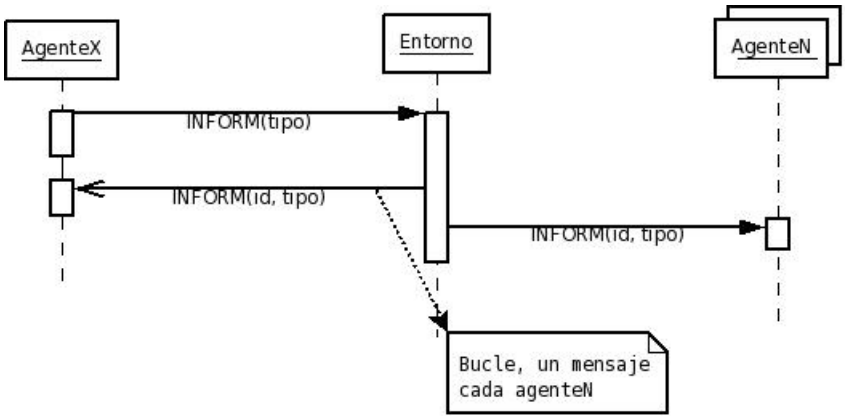

Figura 3.1.4: Protocolo NaceAgente

Por otro lado, al mismo tiempo que se aplicaban optimizaciones, se detectaron algunos errores en el código que fueron subsanados. Algunos de estos errores tocaban aspectos más de diseño que de implementación, como la falta de control para que los agentes no salieran de los límites de su entorno, mecanismo incorporado en los entornos Sabana 3D y Ghost Squadron.

En los aspectos relativos a complejidad, las clases que ofrecieron más margen de mejora fueron las que se encargaban de los recorridos en las listas de creencias e intereses. Se

Nombre del Comportamiento: organizator	
Agente	AgenteSimple
Protocolos	Ninguno
action	Es el encargado de sacar una acción del scheduler y mandarla al actuador del agente siempre y cuando se cumpla la precondición de ejecución de dicha acción. Si no se cumple la precondición de la acción, se eliminan todas las acciones del plan.
onStart	Nada
onEnd	Nada
done	Este comportamiento es cíclico (no termina nunca)

Tabla 3.1: Diseño del Comportamiento “organizator”

consiguió pasar de complejidad lineal a constante en muchos de los métodos que trabajaban con las listas en las clases antes mencionadas y también en modificaCreencia y en modificaInteres. En las clases RelCreencia y RelCreenciaInteres se fusionaron métodos de funcionalidad similar para facilitar la comprensión al desarrollador.

3.1.3.2. Actuador

En este apartado se explicarán los cambios realizados en la comunicación Agente -> Entorno para adecuarlos al modelo de Actuador propuesto en [Imbert, 2005] y del que se habló brevemente en la sección 2.1. En este modelo se propone que los Actuadores deben ser los componentes encargados de ejecutar los operadores contenidos en las acciones sobre el entorno.

El componente Actuador desarrollado en la primera versión de COGNITIVA era una clase, cuyo método principal se encargaba de modificar las creencias del agente en función de la acción enviada al Entorno. Seguidamente, la acción era enviada al entorno dentro del comportamiento organizator. Este proceder tenía dos problemas serios:

- En primer lugar, las creencias del agente eran modificadas por la clase Actuador antes de haber sido enviadas al Entorno, por lo que siempre se contaba con que la acción se llevaba a cabo satisfactoriamente y con los valores supuestos (aunque luego el Entorno los corrigiera). Además la clase Actuador realizaba así tareas del intérprete, como son la modificación de las creencias como resultado de los cambios en el entorno.
- En segundo lugar, al enviar las acciones al Entorno en el comportamiento organizator, se producía comunicación entre el módulo cognitivo y el Entorno, saltándose el paso por los actuadores, componente que por tanto no estaba desarrollado.

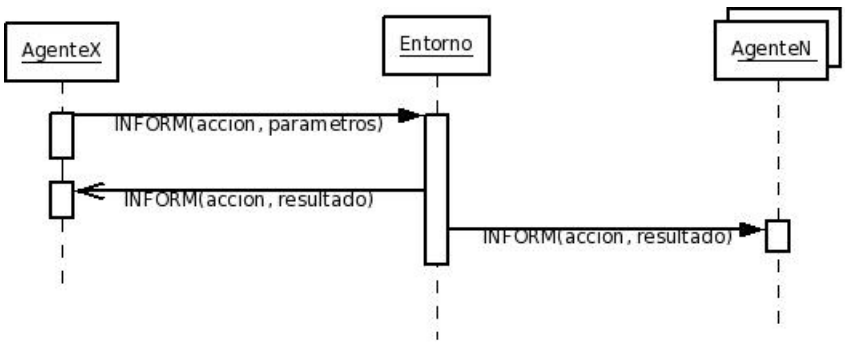

Figura 3.1.5: Protocolo CambioAgente

Para solucionar esta situación, se rediseñó la clase Actuador, convirtiéndose en un paquete con dos comportamientos que se agregan al agente: uno que comunica al Entorno el nacimiento del agente, y otro para enviar al Entorno las acciones que va realizando el agente. El comportamiento organizator coloca, en una zona de memoria compartida entre los comportamientos, la próxima acción a ejecutar; el comportamiento de Actuador la recoge y, tras un procesamiento, la envía al Entorno. De esta forma se desacopla el módulo cognitivo del Entorno y el Actuador pasa a ocuparse de su verdadera labor.

Además, la modificación de las creencias como resultado de las acciones, fue trasladada al comportamiento esperaCambio, que recoge las acciones ejecutadas por el resto de agentes y el propio, y modifica las creencias e intereses, realizando las tareas del intérprete. En la figura 3.2 se muestra el diagrama de comportamiento de esperaCambio de una Concreción Contextual en particular, Sabana 3D. También se recogen los diagramas de los comportamientos expectativasTiempo y nuevaIdentificacion en las figuras 3.3 y 3.4 respectivamente.

3.1.4. Tipos de Datos

Para concluir la sección de refactoring se explicarán las mejoras introducidas en lo relativo a los tipos de datos manejados por la arquitectura. Estas novedades surgen de las necesidades ya expresadas en [Leal, 2005, Molina, 2005] de ampliar los tipos de valores.

3.1.4.1. Interfaz Valor

Los tipos de datos en COGNITIVA estaban definidos en una jerarquía donde la clase padre era Valor y los diferentes tipos de datos eran hijos de esa clase. De esta manera para añadir nuevos tipos de valores bastaría con declarar la nueva clase y definirla como clase hija de la clase Valor.

La ventaja de este modelo es que permite trabajar con los valores de las creencias e intereses ignorando el tipo de valor definido; por ejemplo, se pueden comparar los valores de dos creencias sin necesidad de conocer el tipo de esos valores. Sin embargo, durante la fase de pruebas del sistema original, se pudo apreciar que el uso de una jerarquía no bastaba para trabajar de forma verdaderamente transparente a los tipos de datos: La clase padre Valor define unos métodos comunes para todos los tipos de datos, pero no obliga a que las clases hijas sobreescriban esos métodos.

Esta peculiaridad exige conocer de antemano que el tipo de datos que se va a manejar tiene definido el método y no va a causar un fallo en la ejecución del sistema. Por consiguiente, se optó por utilizar un esquema más rígido que el de jerarquía para implementar los tipos de datos, una interfaz. La interfaz Valor define las cabeceras de todos los métodos que han de sobreescribir obligatoriamente todas las clases que implementan a la interfaz, y de esta manera, todo el código de la arquitectura asume la implementación de dichos métodos, abstrayéndose completamente del tipo de valor que se define. En las figuras 3.1.6 y 3.1.7 se pueden observar las diferencias entre el esquema de jerarquía y el esquema de interfaz de los tipos de datos.

Nombre del Comportamiento: esperaCambio	
Agente	Cebra/León
Protocolos	CambioAgente
action	Se recibe del entorno los datos correspondientes a los cambios que se han producido en el mismo. Las acciones que se pueden recibir y lo que se hace es: ▪ **"nueva posición agente":** Tanto si el que cambia de posición es el propio agente como cualquier otro agente en el sistema, se recalculan las distancias entre los agentes. Si alguno de los otros agentes pasa a verse, entonces se generan las actitudes pertinentes. Si alguno de los agente deja de verse entonces dejan de tenerse en cuenta las actitudes sobre dichos agentes. Si el que se mueve es el propio agente, se recalcula el valor del cansancio en función de lo que se haya movido el agente. ▪ **"beber":** Se recalcula el valor de la sed del propio agente en función de la cantidad bebida. ▪ **"comer":** Se recalcula el valor del hambre del propio agente en función de la cantidad comida.
onStart	Nada
onEnd	Nada
done	Este comportamiento es cíclico (no termina nunca)

Tabla 3.2: Diseño del Comportamiento "esperaCambio"

Nombre del Comportamiento: expectativasTiempo	
Agente	AgenteSimple
Protocolos	Ninguno
action	Primero evalúa si se cumple alguna de las expectativas generadas por la acción concreta que ha realizado el agente. Pueden darse los siguientes tres casos: ▪ Si alguna de las expectativas se cumple, se aplican las modificaciones sobre las emociones. ▪ Si alguna de las expectativas caduca, se elimina el plan. ▪ Si no ocurre ninguno de los casos anteriores, no se hace nada. Después se hace los mismo con las expectativas generadas por alguna acción abstracta.
onStart	Nada
onEnd	Nada
done	Este comportamiento es cíclico (no termina nunca)

Tabla 3.3: Diseño del Comportamiento “expectativasTiempos”

Nombre del Comportamiento: nuevaIdentificación	
Agente	Cebra/León
Protocolos	NuevoAgente
action	Se reciben los datos, enviados por el entorno mediante la performativa INFORM, de un nuevo agente en el sistema. Si el nuevo agente es el propio agente que recibe el mensaje, se actualiza la posición del agente en el entorno. Si por el contrario, los datos se corresponden con otro agente, se alamacena la posición de este nuevo agente en el sistema y se calcula la distancia que hay con él, para saber si le tenemos en nuestro entorno de visión.
onStart	Nada
onEnd	Nada
done	Este comportamiento es cíclico (no termina nunca)

Tabla 3.4: Diseño del Comportamiento “nuevaIdentificación”

3.1.4.2. Nuevos Tipos

Una vez adoptado el esquema de interfaz señalado en el anterior punto, se procedió a incorporar los nuevos tipos que surgieron como necesidad según se iban desarrollando los

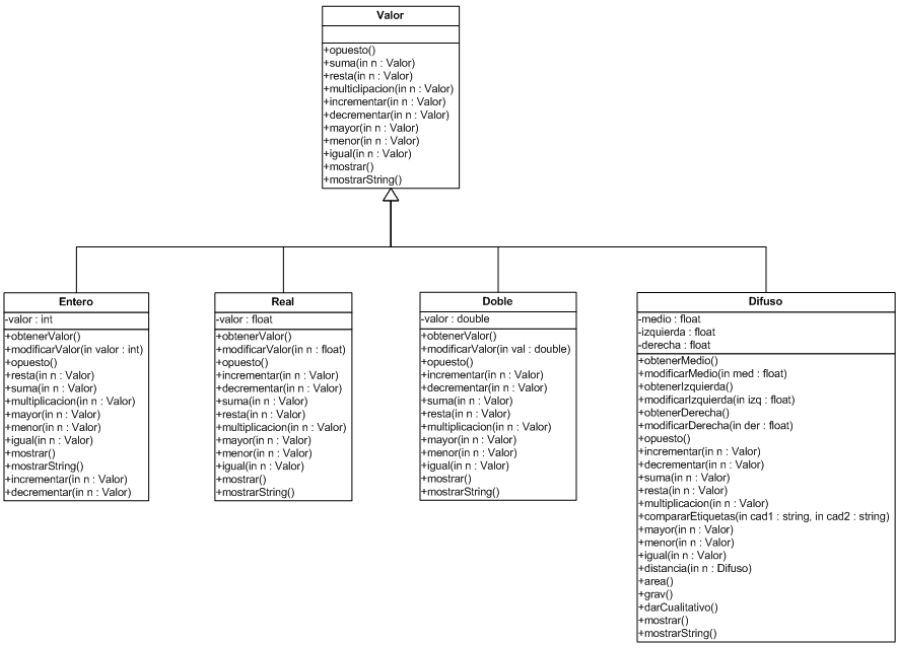

Figura 3.1.6: Jerarquía Tipos de Datos

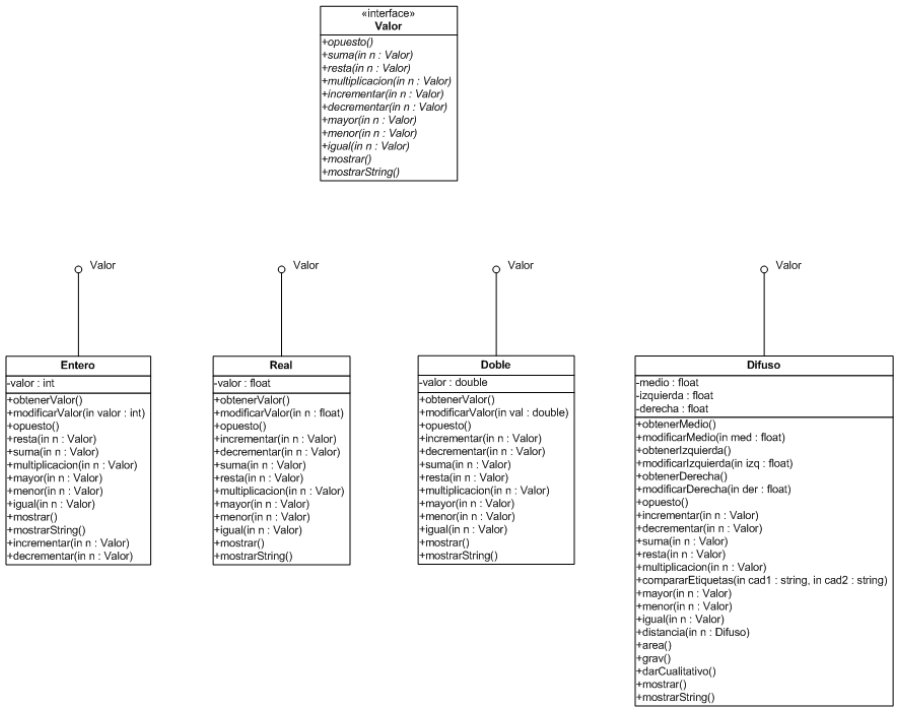

Figura 3.1.7: Interfaz Tipos de Datos

objetivos del presente trabajo.

El primero de estos tipos fue el tipo Booleano, necesario para manejar objetos lógicos en forma de semáforos, como pueden ser los de *acción ejecutada* o *acción de mayor prioridad*. Como ya se ha mencionado, el tipo Booleano al implementar la interfaz Valor, debe sobreescribir métodos como *sumar*, *restar*,... los cuales se adaptaron para dar un sentido lógico que no produjera un resultado incorrecto.

Además del tipo Booleano, surgió la necesidad de utilizar objetos Cadena como tipos de datos a consecuencia del nuevo tipo de creencias definido en la sección 3.5. Estos objetos Cadena representan una cadena de caracteres, y redefinen los métodos de Valor además de incorporar los propios.

3.1.4.3. Serialización

En la implementación original de COGNITIVA surgió la necesidad de definir clases propias para los tipos de datos, debido fundamentalmente a que uno de los tipos de datos que debían soportar las creencias y los intereses era el tipo Difuso, un valor de lógica borrosa que se expresa con una representación simplificada a modo de triángulo: punto medio, extremo izquierdo y extremo derecho. De esta manera, dado que el resto de tipos tenía que operar con el tipo Difuso, se optó por definir todos de manera que se permitiera la interacción de cualquier tipo con otro siguiendo el esquema de interfaz explicado en el primer apartado.

Sin embargo, en la primera versión, cuando surgía la necesidad de enviar alguno de estos objetos Valor a otro agente, a través del RMI de Jade, se extraía su contenido y se encapsulaba en un objeto de las clases de tipos predefinidos en Java: *Double, Integer o Float*. Como no había la necesidad de enviar objetos Difusos, este sistema funcionaba, pero exigía encapsular los objetos en el envío y posteriormente desencapsularlos en la recepción.

Para conseguir enviar por el RMI los objetos de tipos de datos definidos en la arquitectura, tan sólo hizo falta declarar esas clases como *serializables* de manera que pudieran ser escritas y leídas por la red. Esto permitió ser realmente independiente de las clases de tipos de datos predefinidos de Java, así como abrir la puerta a la posible necesidad futura de enviar objetos de tipo Difuso a otros agentes.

3.1.4.4. Tipo Difuso

Por último se señalan dos reformas adoptadas en la clase Difuso, para conseguir un comportamiento más real:

- Los tipos Difusos también pueden expresarse como la etiqueta borrosa a la que más se parece su triángulo de valores : *nada, ligeramente, medianamente, bastante y absolutamente*. Para conseguir un resultado más real y mejorar la precisión en el comportamiento del agente se sustituyó la antigua comparación entre borrosos, que se basaba en la etiqueta más cercana (por lo que dos difusos que se parecieran a *bastante* eran iguales), por una comparación basada en el centro de gravedad (en nuestro caso, el punto medio). Así un valor difuso sólo es igual a otro si son iguales sus centros de gravedad, en caso contrario, uno es mayor que el otro. Por supuesto, esto es una simplificación, pero para el tipo de razonamiento que se realiza con los números borrosos, es válida. Este cambio es especialmente importante en los justificadores de reglas que utilizan umbrales de intereses, dado que se disparan si el valor de la creencia supera o está por debajo del umbral señalado. Con la comparación basada en etiquetas el umbral tardaba mucho en ser superado e incluso no lo era nunca para etiquetas como *bastante*.

- La otra reforma adoptada es garantizar que los extremos de un valor difuso no se distancien más de una determinada cantidad, debido a que por encima de ésta la base del triángulo se vuelve demasiado ancha y la comparación con etiquetas resulta engañosa. Así, tras cualquier operación sobre un difuso, se corrigen sus extremos de forma que la suma aritmética del extremo izquierdo más el extremo derecho no supere nunca el valor real *0,4*.

3.2. Ficheros de configuración

El objetivo de COGNITIVA, como arquitectura genérica, es permitir que cualquier desarrollador que requiera un entorno multiagente de base emocional, pueda construir uno rápidamente y con poco esfuerzo. Sin embargo, uno de los escollos con los que choca el desarrollador es el confuso formato de los ficheros de configuración del agente, más cercano a las matemáticas que al lenguaje natural. En la primera versión, los ficheros de configuración eran poco más que ristras de números y valores booleanos que, en función de su posición, significaban una cosa u otra.

Por este motivo se planteó como necesidad, ya señalada en [Leal, 2005, Molina, 2005], hacer estos ficheros más intuitivos para el usuario, evitando que tuviera que consultar el código fuente para saber qué debe escribir.

El formato planteado es muy cercano al de la especificación formal, sobre todo en la definición de las acciones y reglas. Se han sustituido los números borrosos por su etiqueta borrosa más cercana y los valores booleanos se han cambiado por los literales que representan. Para ilustrar esta mejora, se muestran las plantillas de los nuevos ficheros de configuración que han de adaptar los desarrolladores a sus concreciones contextuales.

Hay que señalar que la figura 3.2.1 representa los valores que puede tomar la etiqueta VALOR presente en el resto de ficheros, se muestra en una figura aparte para facilitar la comprensión. Las plantillas expresan las alternativas como corchetes [] y las repeticiones con puntos suspensivos ....

```
difuso:[nada | ligeramente | medianamente | bastante | absolutamente]
real:valor_real
entero:valor_entero
doble:valor_doble
cadena:valor_cadena
umbral:[inferior | superior]
```

Figura 3.2.1: Expresión VALOR

3.2.1. Acción

La figura 3.2.2 es la plantilla del fichero *accion.txt*. Se establece un fichero accion.txt por cada acción contextual que utilicen los agentes, por tanto su localización física se sitúa en el paquete correspondiente a cada acción, como se definió en la sección 3.1.2.

En la anterior versión, accion.txt contenía la configuración de todas las acciones que hacía el agente además de las reglas y las reglasMetas. Las reglasMetas son reglas que, en lugar de disparar acciones, lanzan metas a alcanzar por el agente. Son la implementación de los mecanismos definidos en la sección 2.8.2. Por tanto se separaron las reglas y reglasMetas de las acciones, añadiendo dos nuevos ficheros a los originales: reglas.txt y reglasMetas.txt. Además se creó el fichero acciones.txt que contenía los nombres de las acciones que podía hacer cada agente para que buscara la configuración de las mismas en los paquetes del mismo nombre. La estructura de una acción es: tipo de acción, nombre, prioridad, número de plan, caducidad, subtipo de acción, precondición, postcondición y expectativas.

3.2.2. Creencias

En la siguiente figura, la 3.2.3, se muestra la plantilla del fichero *creencias.txt*. En este fichero se localizan las creencias iniciales del agente, por tanto se localizan en el paquete *agente*_estructura. Se definen las creencias del modelo personal, el resto de creencias, los intereses y las relaciones entre los componentes del modelo personal.

```
[concreta | abstracta] nombre_accion prioridad:valor_entero plan:valor_entero caducidad:valor_entero [externa | interna]
Pre
atributo(objeto) operador VALOR
...
Post
atributo(objeto) operador VALOR
...
Expectativas
Expectativa [deseo | nodeseo] [expectacion | noexpectacion]
atributo(objeto) operador VALOR
...
finexpectativa
Expectativa [deseo | nodeseo] [expectacion | noexpectacion]
atributo(objeto) operador VALOR
...
finexpectativa
...
finaccion
```

Figura 3.2.2: Fichero accion.txt

```
Personalidad
atributo(objeto) VALOR
...
Emociones
atributo(objeto) VALOR degradado:velocidad_degradado(real)
...
Estados Fisicos
atributo(objeto) valor VALOR reposo VALOR degradado:velocidad_degradado(real)
...
Actitudes
atributo(objeto) VALOR
...
Creencias
atributo(objeto) VALOR
...
Intereses
atributo superior:VALOR inferior:VALOR
...
RelacionesCreencias
influyente:atributo influida:atributo [aumenta|disminuye] VALOR
...
PersoActitudes
influyente:atributo influida:atributo [aumenta|disminuye] VALOR
...
ActitudEmociones
influyente:atributo influida:atributo [aumenta|disminuye] VALOR
...
CreenciasIntereses
influyente:atributo influida:atributo [aumenta|disminuye] [superior|inferior] VALOR
...
EstadoEmociones
influyente:atributo influida:atributo [aumenta|disminuye] VALOR
```

Figura 3.2.3: Fichero creencias.txt

3.2.3. Reglas

Se ha señalado que las reglas de cada agente se han situado en un nuevo fichero, *reglas.txt*. En la figura 3.2.4 encontramos una plantilla del mismo. También se sitúa en el paquete *agente*_estructura. Su estructura es: antecedente, prioridad, acciones a disparar y parámetros de las acciones.

```
regla
atributo() operador VALOR
...
finantecedente
prioridad:valor_entero
accion:nombre_accion
accion:nombre_accion2
...
Parametros
Parametro [entero | doble | real | difuso | cadena] parametro
...
finregla
```

Figura 3.2.4: Fichero reglas.txt

3.2.4. ReglasMetas

Análogamente a las reglas, las reglasMetas también tienen su propio fichero dentro del paquete *agente*_estructura, el fichero *reglasMetas.txt.* La figura 3.2.5 es una plantilla del formato de dicho fichero. Cada reglaMeta tiene un antecedente, prioridad, caducidad y un estado meta que desea alcanzar.

```
reglameta
atributo() operador VALOR
...
finantecedente
prioridad:numero_entero
caducidad:ticks
atributo() operador VALOR
...
finmeta
```

Figura 3.2.5: Fichero reglasMetas.txt

3.2.5. Acciones

Una de las características más interesantes que se han añadido a COGNITIVA es la capacidad de agregar acciones contextuales a la arquitectura de forma totalmente *transparente* al resto de componentes; esta capacidad apareció como consecuencia directa tras haber determinado que cada tipo de acción será una clase (ver sección 3.1.2), en lugar de un objeto como se hacía en la primera versión. Para agregar una acción basta con crear su

paquete correspondiente dentro del paquete Accion y añadir su nombre en el fichero *acciones.txt* (ver figura 3.2.6) de los agentes que podrán realizar esta acción.

El lector de acciones instanciará la clase correspondiente gracias a la capacidad de JAVA de instanciar clases de forma reflexiva, esto es, a partir del literal que corresponde al nombre de la clase, sin necesidad de estar explícitamente escrito en el código. Creemos que poder agregar acciones al modelo sin necesidad de compilar el resto del código supone una importante mejora en la plasticidad de la arquitectura.

```
nombreAccion1
nombreAccion2
nombreAccion3
...
```

Figura 3.2.6: Fichero acciones.txt

3.2.6. Tabla

Finalmente, el último fichero de configuración que resta comentar es el *tabla.txt* (figura 3.2.7) que recoge las relaciones entre las expectativas y las emociones del agente. Tiene una línea por cada posible combinación entre el tipo de expectativa (esperada y/o deseada) y el estado de la expectativa (cumplida, no cumplida o no confirmada), lo que suman doce combinaciones. Tras cada combinación se listan las emociones afectadas con su grado (el signo del grado indica el modo). Al haber determinado una tabla por acción (ver sección 3.1.2) el fichero se localiza en el paquete correspondiente a cada acción.

3.3. Expectativas

Durante el largo periodo en el que se realizaron pruebas exhaustivas para analizar el comportamiento de COGNITIVA, se observó que el componente que más influencia tenía en los resultados eran las expectativas.

Las expectativas, como ya se definió en la sección 2.5, modifican las emociones del agente en función del resultado de las mismas, esto es: cumplidas, no cumplidas o no confirmadas. A su vez los valores de las emociones tienen su papel en los justificadores de las reglas reactivas, por lo que la modificación de las emociones puede dar lugar a que se disparen unas acciones u otras.

Partiendo de éstas bases, la causa de la desmedida influencia de las expectativas en el comportamiento del agente, es la abundancia de lugares en los que estas expectativas

afectan a las emociones y la desproporcionada manera en que lo hacen. Es probable que se deba a que en [Imbert, 2005] no se profundiza demasiado en las expectativas, y éstas quedaron definidas muy libremente para los programadores.

En las siguientes subsecciones analizaremos las reformas realizadas en las expectativas para conseguir un comportamiento más real y coherente.

3.3.1. Expectativas no confirmadas

Cuando una expectativa se crea, es decir, cuando la acción que la originó se manda al actuador, su estado es *no confirmada*. Esto quiere decir que aún no se sabe si la expectativa se ha cumplido o no, por tanto permanece en ese estado hasta que se cumple, o hasta que se tenga constancia de que no se cumplirá.

En [Imbert, 2005] la forma en que afectan las expectativas no confirmadas a las emociones es modificando el valor de las mismas en el momento en que la acción pasa al actuador. Sin embargo, el efecto buscado con las expectativas no confirmadas es el de mantener un cierto estado de las emociones hasta que se averigüe si se han cumplido o no y, modificando tan solo su valor, sólo se consigue este efecto al principio; según pasa el

```
0. cumplida, esperada, deseada
accion:emocion grado:numero entre -5 y 5
...
1. cumplida, esperada, nodeseada
accion:emocion grado:numero entre -5 y 5
...
2. cumplida, noesperada, deseada
accion:emocion grado:numero entre -5 y 5
...
3. cumplida, noesperada, nodeseada
accion:emocion grado:numero entre -5 y 5
...
4. nocumplida, esperada, deseada
accion:emocion grado:numero entre -5 y 5
...
5. nocumplida, esperada, nodeseada
accion:emocion grado:numero entre -5 y 5
...
6. nocumplida, noesperada, deseada
accion:emocion grado:numero entre -5 y 5
...
7. nocumplida, noesperada, nodeseada
accion:emocion grado:numero entre -5 y 5
...
8. noconfirmada, esperada, deseada
accion:emocion grado:numero entre -5 y 5
...
9. noconfirmada, esperada, nodeseada
accion:emocion grado:numero entre -5 y 5
...
10. noconfirmada, noesperada, deseada
accion:emocion grado:numero entre -5 y 5
...
11. noconfirmada, noesperada, nodeseada
accion:emocion grado:numero entre -5 y 5
...
```

Figura 3.2.7: Fichero tabla.txt

tiempo el valor de las emociones vuelve a su estado original, o se ve afectado por otras circunstancias.

La única manera de mantener el estado hasta conocer el resultado es modificando el valor de reposo de la emoción en lugar de sólo el valor. Así en la nueva versión de COGNITIVA las expectativas no confirmadas modifican el valor de reposo de las emociones a las que afectan en el momento en que la acción se manda al actuador, y cuando se conoce el resultado (cumplida o no) se deshace la modificación, volviendo al valor de reposo original.

3.3.2. Comprobación de Expectativas

Se ha señalado que el principal problema de las expectativas es su excesiva influencia en el comportamiento del agente, debido a la gran cantidad de veces que afectan a las emociones.

El primer paso fue, por tanto, identificar las zonas del código en las que se comprueban y aplican las expectativas. Se detectaron dos fragmentos:

- En el comportamiento *expectativasTiempo* que las analiza cada segundo.
- Cada vez que se modifica una creencia, se observa si como resultado se ha cumplido alguna expectativa.

Se observó que el evaluar las expectativas cada vez que se modifica una creencia o interés producía un comportamiento irreal en el agente. Se explica con un ejemplo:

Si el agente *piloto* lanza la acción *eludirEnemigo*, una de sus expectativas (deseada y esperada) es no estar cerca del enemigo; si seguidamente el enemigo se *mueve* y se evalúa la expectativa (al estar la creencia de la *distancia* al enemigo relacionada con ella), se cumplirá la expectativa y se modificarán las emociones en consecuencia (posiblemente aumentará el *miedo* del agente). Sin embargo, esto no debería suceder ya que no ha pasado suficiente tiempo para que puedan apreciarse los efectos de *eludirEnemigo*.

Basándonos en lo explicado en el ejemplo, se tomó la decisión de eliminar la comprobación de las expectativas en la modificación de creencias o intereses para garantizar que pasa un tiempo entre el lanzamiento de la acción y la comprobación de expectativas. En la nueva versión por tanto, sólo se evalúan las expectativas en el comportamiento *expectativasTiempo*.

3.3.3. Eliminación de Expectativas

Finalmente, además de la comprobación antes de tiempo de las expectativas, se detectaron incoherencias y lagunas en lo relativo a las diferentes circunstancias en las que se eliminan las expectativas, y las consecuencias de su eliminación.

En primer lugar, en la versión inicial cuando una expectativa se cumplía o caducaba, se eliminaban el resto de expectativas de la misma acción, dado que el tipo de expectativas propuesto en [Imbert, 2005] son incompatibles entre sí, si se cumple una es imposible que se cumpla la otra. Este proceder es correcto, pero el problema radicaba en que además de eliminarlas, aplicaban su influencia sobre las emociones como caducadas (no cumplidas). Por esta razón, las emociones se modificaban tanto como resultado del cumplimiento o caducidad de la expectativa, como por la caducidad del resto, provocando una variación desproporcionada e incoherente en las emociones.

En la nueva versión, cuando se cumple o caduca una expectativa, se eliminan las expectativas de la misma acción, pero no afectan como no cumplidas. Este problema se daba en las expectativas de las acciones enviadas al actuador, pero para las acciones abstractas los problemas eran mayores.

Una acción abstracta tiene expectativas al igual que una acción concreta: éstas se introducen en el conjunto de expectativas cuando se descompone la acción, y conviven tanto con las expectativas de la acción en el actuador como del resto de acciones abstractas descompuestas. El problema detectado fue en la misma fase que el anterior, pero de una gravedad superior: Al cumplirse o caducar una expectativa proveniente de una acción abstracta, el resto de expectativas de la acción *no* se eliminaban, y, por tanto, permanecían en el conjunto de expectativas a la espera de cumplirse o caducar, aunque esto ya no tuviera sentido.

En esta nueva versión se eliminan las expectativas asociadas a una acción abstracta cuando alguna se cumple o caduca sin, por supuesto, influenciar a las emociones. Finalmente, el último problema acerca de la eliminación de expectativas afectaba a la cancelación de planes.

En COGNITIVA cuando un plan es eliminado, por la razón que sea, se eliminan del organizador todas las acciones pertenecientes a ese plan. También se eliminan las expectativas de la acción en el actuador y se pasa a ejecutar la siguiente acción más prioritaria, que pertenecerá a un plan distinto. Sin embargo, los programadores olvidaron que también se debían eliminar las expectativas abstractas de las acciones ya descompuestas y que pertenecían al plan eliminado; dado que estás ya carecen de sentido al haberse desechado su

plan.

Incorporar este mecanismo fue la última de las reformas emprendidas hasta ahora en el campo de las expectativas, sin embargo se ha revelado como un área de una gran complejidad y en la que seguramente caben estudios que profundicen más que el presente.

Por último señalar que en [Cabeza, S.N.] se recogen diversas pruebas y ejemplos sobre el entorno de Sabana 3D donde se aprecia la influencia de las expectativas sobre las emociones del agente.

3.4. Campo de Visión

En las concreciones contextuales desarrolladas con COGNITIVA es habitual que los agentes cuenten con una representación física, es decir, se encuentren en un lugar y ocupen un espacio. Por tanto, dentro de esta dimensión espacial, uno de los componentes que ofrece más juego es el llamado *campo de visión* del agente; es decir, contar con que el agente no ve todo lo que hay en el entorno, sino sólo lo que está dentro de su campo de visión.

En la primera versión apenas se contaba con esta particularidad que depende, como ya se ha mencionado, de las características de la Concreción Contextual. Sin embargo, en esta versión se han aportado algunas mejoras que afectan principalmente a la concreción Sabana 3D. Ésta consiste en un entorno virtual que simula una sabana africana poblada por leones y cebras virtuales autónomos que realizan sus principales labores para asegurar su supervivencia.

Antes de proceder a cualquier modificación de la gestión del campo de visión se decidió corregir el alcance de la vista en los agentes de forma que se adecuara más al valor real, de esta manera, se aumentó el alcance de la vista de los agentes *cebra*.

Seguidamente empezamos a considerar que los agentes sólo deberían reaccionar a los eventos del entorno cuando éstos se situaran dentro de su campo de visión (salvo casos especiales) y, más aún, tener en cuenta sólo a los objetos e individuos que se pueden percibir. Así, concretando esta idea, establecimos que la actitud *temor* frente a un *león* sólo debe afectar cuando éste está dentro del campo de visión de la *cebra*: Si una *cebra* o un *león* se mueven y pasan de estar *no vistos* a *vistos*, la actitud empieza a afectar a la *cebra* porque el objeto de esa actitud (el individuo *león* en concreto) está presente, y así, se actualiza el valor de las emociones influenciadas por esta actitud. Por el contrario, cuando se pasa de *visto* a *no visto*, se ha de dejar de tener en cuenta la actitud *temor*. Sin

embargo, para obtener un comportamiento realista no basta con modificar el valor de las emociones.

Cuando el león aparece en el campo de visión de la cebra, se comienza a tener en cuenta la actitud de temor hacia él y, por tanto, ésta actualiza el valor del *miedo* que es una de las emociones afectadas por dicha actitud. Ahora bien, supongamos que el león permanece en el campo de visión bastante rato, y, aunque la cebra debería de permanecer con el mismo nivel de tensión, el miedo está disminuyendo debido a que tiende hacia su valor de *reposo*, que en el caso de las emociones depende de su personalidad, aunque suele ser bajo.

Por tanto, para asegurar un comportamiento realista se ha de actualizar también el valor de reposo de las emociones afectadas por la actitud, dado que la cebra ha de tender hacia un nivel alto de miedo durante todo el tiempo que tenga temor hacia el león, es decir, todo el tiempo que *le vea*. Por supuesto, esta modificación del valor de reposo debe deshacerse cuando desaparece la actitud, una vez más, cuando desaparece el león del campo de visión.

De esta forma se incorpora una nueva relación más en el modelo personal entre las actitudes y el valor de reposo de las emociones.

Por último, otra mejora que tiene que ver con el campo de visión de los agentes es que, cuando un agente sale del campo de visión de otro, éste ultimo guardará información sobre su última posición conocida (*creenciaultimax y creenciaultimay)* en lugar de manejar su posición real como se hacía hasta ahora. Cuando vuelva a verle, actualizará el valor y manejará la posición real.

Este tipo de creencias son muy importantes en el nivel deliberativo, donde es necesario que la posición que se tenga de los individuos no vistos no sea la real, para dar pie a que los planes no salgan bien a veces. Es la incertidumbre en el conocimiento de los agentes la que dota de realismo a su comportamiento.

3.5. Creencias Situación Actual

Cuando se definieron las estructuras de información del agente, se mencionó que un tipo de creencias serían las creencias intangibles que el agente mantiene sobre la situación actual, en la sección 2.2.4. Sin embargo, en la primera versión de COGNITIVA este concepto no fue apenas desarrollado, posiblemente porque la gama de tipos de datos manejada no permitía soportar este tipo de creencias. Tan sólo se hizo una ligerísima aproximación con creencias *ad hoc* como la creencia *huyendo(yo)* que tomaba los valores 0 o 1 en función de si el agente estaba o no en situación de huida.

Tras el desarrollo de la nueva clase CreenciaEstado que daba soporte a creencias cuyo valor, en lugar de ser borroso o numérico, fuera un literal de texto, ya estaba listo el cimiento para manejar este tipo de creencias.

Así, tras un análisis de las necesidades de los agentes en distintos contextos, se establecieron dos tipos de creencias estado.

3.5.1. Estado Interno

Un primer tipo de creencias sobre la situación actual son las creencias sobre el *estado interno* del agente, por ejemplo, si el agente se encuentra *andando* o *corriendo.* Es un conocimiento que el agente necesita tener de sí mismo, para poder garantizar la coherencia y el realismo del comportamiento del agente. El agente puede tener tantas creencias sobre su estado interno como necesite (en teoría tantas como estados en los que pueda encontrarse simultáneamente). En estas creencias el objeto siempre es *yo*, es decir, el propio agente, y el valor es el literal con el nombre del estado en que se encuentra el agente.

3.5.2. Estado de la Comunicación

El segundo tipo son las creencias sobre el *estado de la comunicación* con otros agentes. Es decir, si nos hayamos en un proceso de negociación con otro individuo, es necesario conocer en todo momento cuál es el estado de dicha negociación para poder determinar que acciones comunicativas proceden en esa fase. Un ejemplo sería que la comunicación se encuentra en estado *esperando_respuesta*. Así, en este tipo de creencias, el objeto nunca puede ser *yo*, siempre ha de ser el nombre del agente con el que se está desarrollando la comunicación. En el capítulo 4 se tratara ampliamente sobre la función de este tipo de estados y su importancia.

3.6. Planificador

La alternativa elegida en [Imbert, 2005] para llevar a cabo el proceso de planificación es el planificador SHOP2 [Ilghami., 2006]. Este planificador es un sistema automático, independiente del dominio y está basado en descomposición de tareas ordenadas. Utiliza un tipo de planificación de red de tareas jerárquicas o HTN, según sus siglas en inglés. Su elección se debe a que se adapta bastante bien a las estructuras propuestas.

Aunque este planificador requiere que los métodos de descomposición de tareas sean dependientes del dominio, es decir, deben ser especificados en la fase de Concreción Contextual, es un planificador bastante rápido y fiable, y es uno de los más utilizados actualmente.

Para esta versión de COGNITIVA se decidió implantar la implementación Java de SHOP2, esto es, JSHOP2. Esta versión es más potente y además al estar desarrollada en Java, al igual que COGNITIVA, permite que se realicen modificaciones para adecuarla a nuestras particularidades, a la vez que se garantiza una total compatibilidad.

3.6.1. Envoltorio Planificador y Conversor

Pese a que JSHOP2 está implementado en Java, su API es bastante compleja, e implantarla directamente en COGNITIVA puede requerir demasiado tiempo en formación para conocer todos sus mecanismos.

Afortunadamente, el planificador JSHOP2 también se utiliza en otro de los proyectos del Laboratorio Decoroso Crespo, ENVIRA, y para este proyecto se desarrolló un envoltorio que encapsula las principales funciones de JSHOP2 (ver [Maján, 2009]) y las ofrece en una API muy intuitiva y que proporciona todas las necesidades requeridas por el proceso de planificación de COGNITIVA. Por tanto, implantar el planificador ha resultado ser una tarea bastante sencilla ya que en todo momento se manejaron objetos Java del envoltorio con métodos documentados.

Aun así fue necesario desarrollar una clase Conversor que sirviera de interfaz entre el envoltorio y COGNITIVA de manera que, a partir de las creencias y la meta del agente, se obtuviera la lista de acciones para lograr la meta.

El proceso de planificación por tanto consta de las siguientes fases:

1. Se codifican en lenguaje JSHOP2 (similar al LISP) las creencias del agente (estado actual) y la meta (estado deseado) en el fichero de problema (problem.txt)

2. Se crea un planificador para el dominio requerido (ver siguiente apartado) con el método c*rearPlanificador* de la clase Planificador.

3. Se recogen los planes elaborados (lista de objetos Plan) con el método *planificar* de la clase Planificador.

4. Si hay al menos un plan, se coge el primero y se decodifica en acciones de COGNITIVA. Para decodificar se hace uso de los métodos del envoltorio para la extracción del nombre y los parámetros de las acciones obtenidas.

3.6.2. Dominio JSHOP2 mejorado (Sabana 3D)

Una vez puesto en marcha el nuevo planificador, tras haberlo probado con la Concreción Contextual Sabana 3D, explicada en [Cabeza, S.N.], se consideró la posibilidad de ampliar el dominio de planificación de dicha concreción para obtener una simulación más realista y de paso familiarizarse con el lenguaje JSHOP2.

En concreto la mejora consistió en optimizar los planes que la *cebra* elaboraba para *beber, comer o guarecerse*, de manera que esquivase al león cuando fuera posible, en lugar de simplemente aumentar su umbral de interés de la emoción *miedo* para evitar *huir* ante la aparición del mismo. Los detalles del nuevo plan y las pruebas realizadas se explican en [Cabeza, S.N.]. Tan sólo señalar aquí que esta tarea supuso un auténtico reto, por ser un lenguaje funcional heredero de LISP, que pudimos realizar gracias a Ramón Recuero, compañero del laboratorio experto en la materia.

3.6.3. Gestor de Metas

Una vez generada una meta, ésta ha de entrar en un proceso de mantenimiento mientras se encuentre en el repositorio global de metas, desde que se introduce, hasta que es satisfecha o es expulsada por dejar de estar vigente.

En la primera versión de COGNITIVA no se realizaba este mantenimiento, pese a estar definido en [Imbert, 2005], una meta era generada cuando se daban las condiciones propicias e inmediatamente se elaboraba un plan para satisfacerla, obteniendo sus acciones e introduciéndolas en el organizador. En este punto dejaba de utilizarse la meta, por lo que conceptos como satisfecha o vigente no se tenían en cuenta.

En esta versión se ha desarrollado un componente, el *gestor de metas* que realiza este mantenimiento. Sus criterios están fundamentados en el proceso de mantenimiento definido brevemente en la sección 2.8.2 y se muestran a continuación. Seguidamente se muestra el diagrama del comportamiento (ver definición en la sección 3.1.3) *gestorMetas* asociado.

- Si una meta ha sido a*lcanzada,* se elimina del conjunto de metas dado que ha terminado su ciclo de vida.

- Si una meta está *en espera* y sigue vigente y sin caducar, se intenta elaborar un plan para cumplirla pasando a e*n proceso.* Si no se obtiene el plan, pasa a estar *inalcanzable*, en cambio si se logra elaborar un plan la meta pasa a estar *planificada*. Seguidamente, al descomponer el plan en acciones e introducirlas al organizador la meta pasa a estar *en ejecución.* Si la meta hubiera dejado de estar vigente o hubiera caducado, habría pasado al estado *cancelada*

- Si la meta está *en ejecución* y deja de estar vigente, o caduca, se elimina el resto de acciones del mismo plan, así como sus expectativas, y la meta pasa a estar *cancelada*.

- Si la meta está en *inalcanzable* y sigue vigente y sin caducar, pasa a estar de nuevo e*n espera*. Si no, pasa a estar c*ancelada.*

- Finalmente, si la meta está *cancelada,* se elimina del conjunto de metas.

Nombre del Comportamiento: gestorMetas	
Agente	AgenteSimple
Protocolos	Ninguno
action	Se dedica al mantenimiento de las metas del agente. Comprueba cíclicamente la vigencia de las metas del agente. Las metas son eliminadas cuando son alcanzadas, cuando el tiempo de caducidad de la meta venciese o por los acontecimientos ocurridos desde su generación, no tuviese sentido el mantenerla.
onStart	Nada
onEnd	Nada
done	Este comportamiento se ejecuta cada segundo

Tabla 3.5: Diseño del Comportamiento “gestorMetas”

3.7. Reglas

La estructura seleccionada en la Concreción Funcional desarrollada [Imbert, 2005] para el procesamiento de reflejos y reacciones conscientes es la de reglas. Las reglas se adaptan a la morfología de las reacciones, contando con un *antecedente* en el que se agrupan disparadores y justificadores, y un *consecuente,* en el que se recogen las acciones respuesta.

Además también se adoptó esta estructura para representar la asociación entre una meta y los eventos que la provocan. El antecedente se compondría de los perceptos, restricciones o acontecimientos pasados que provocan la meta y el consecuente sería la meta generada.

Sin embargo, al realizar las primeras pruebas para examinar el funcionamiento de la arquitectura, se observaron ciertas limitaciones o características mal diseñadas de las reglas que provocaban un comportamiento poco acorde con lo esperado y sobre todo distinto de lo que ocurriría en un escenario real. En las siguientes subsecciones se explican los problemas encontrados y las soluciones adoptadas.

3.7.1. Lanzamiento de reglas

La característica que provocaba un comportamiento más irregular era el lanzamiento de una regla aunque ya hubiera sido lanzada y su consecuente se encontrara aún el organizador. El mecanismo implementado funcionaba de manera que, cuando cambiaba una creencia o interés que afectara al antecedente de una regla, éste se evaluaba, y si era cierto se disparaba la regla. Por tanto, si en un antecedente cambiaban varias creencias, éste se evaluaba todas las veces y se disparaba también todas las que se cumplían.

La consecuencia era que se quedaban en el organizador acciones que ya no tenía sentido realizar, debido a que se habían realizado ya la primera vez que se lanzó la regla. En principio, esto no resultaba un problema muy grande ya que, si en el momento de ejecutar la acción la precondición no se cumplía, ésta no se ejecutaba. Sin embargo, se daba con frecuencia el caso de que la acción siguiera cumpliendo la precondición, pero las condiciones que la lanzaron ya no estaban vigentes, dado que el antecedente de la regla y la precondición no tienen porque ser iguales.

Así, cuando las acciones duplicadas se ejecutaban producían un comportamiento incoherente con la situación en ese momento de la regla. Por ejemplo, si un agente que represente a un *piloto de avión* tiene que huir de un *barco* y por tanto dispara la regla que provoca que *ascienda* para esquivarlo, no tendrá sentido que una vez ascendido continúe ascendiendo debido a que acciones duplicadas se introdujeron en el organizador. Es más, es posible que el agente en ese momento ya no tenga que *ascender* sino *descender* porque en ese momento la amenaza es de un *caza* y por tanto se mezclen las acciones *descender* nuevas con las *ascender* generadas por la anterior regla produciendo un comportamiento incoherente. Además, estas acciones introducían sus expectativas, que interferían a su vez en las emociones del agente provocando un comportamiento más extraño aún.

La manera de solucionarlo ha consistido en asociar las acciones a la regla o reglaMeta que la generó, según proceda. Por tanto, cuando se dispara una regla, ésta se bloquea, y no se desbloquea hasta que no se da alguna de estas dos situaciones:

- Se cancela el plan que la generó, por tanto se eliminan las acciones que queden en el organizador procedentes de ese plan, y se desbloquea la regla (al eliminar la última acción del plan).
- Se ejecuta la última acción del plan que generó, por tanto, tras ejecutar la acción se desbloquea la regla asociada a dicha acción.

3.7.2. Parámetros para las acciones

Además de características mal diseñadas, surgió la necesidad de incorporar nuevos mecanismos para paliar algunas limitaciones. Este es el caso de incorporar el uso de parámetros para las acciones.

En la primera versión, los parámetros para las acciones eran fijos, es decir, venían escritos en el código y tan sólo se alteraban con la corrección realizada en el método parametrosDinamicos (ver sección 3.1.2) que modificaba ligeramente el valor de los parámetros fijos en función de las emociones y estados físicos del agente.

En esta versión se optó por, manteniendo la corrección de parametrosDinamicos, ofrecer al usuario la posibilidad de especificar los parámetros de las acciones desde los ficheros de configuración. Debido a que la definición de las acciones es común para todos los agentes (*mover* es igual tanto en *cebra* como en *león*), el único lugar donde podían especificarse era en los ficheros de reglas, que eran diferentes para cada agente.

De esta manera en los ficheros de reglas reactivas (reglas.txt) tras cada una de las acciones que lanza la regla se pueden definir tantos parámetros como se quiera y del tipo que sea necesario, dentro de los tipos definidos en el paquete Tipos Datos (ver sección 3.1.4). Luego, estos parámetros se pueden procesar en el código de cada clase Accion (atributo parámetros) y son enviados al entorno en el contenido de la acción, inmediatamente después de el nombre de la misma.

3.7.3. Acciones Abstractas Recursivas

Finalmente, a la vez que se profundizó en el estudio del lanzamiento repetitivo de reglas, se identificaron algunos tipos de reacciones que se lanzaban sucesivamente, aún contro-

lando que no se lanzara una hasta que no terminara la otra. Estas reacciones por tanto, respondían a situaciones que permanecían presentes a medio o incluso a largo plazo y que precisaban de un tratamiento más específico que la mera sucesión de lanzamientos de la misma regla.

Así, se identificaron reacciones como *cazar* o *huir* que permanecían vigentes durante todo el periodo que el *cazador* o la *presa* permanecían en el ámbito de visión. Estas reacciones por tanto, cambiaban el estado interno del agente (ver la sección 3.5) ya que el agente debía estar *huyendo* o *cazando* un largo periodo de tiempo.

La manera en que se decidieron implementar estas reacciones fue con la estructura de *acciones abstractas recursivas*. Esta estructura, implantada en los consecuentes de las reglas que las disparan, se compone de:

- Una acción interna *empiezaX* que requiere que el estado interno del agente sea distinto de X y que lo único que hace es cambiar el estado interno del agente a X.
- Una acción abstracta X que al razonarse se descompone en la(s) accion(es) concreta(s) que se precise(n) y en ella misma. Por ejemplo, *eludirEnemigo* se descompone en *girarIzquierda* o *girarDerecha* según proceda, y de nuevo en *eludirEnemigo*.
- Una acción interna *terminaX* que se ejecuta al dejar de darse las condiciones para seguir en estado X, es decir, cuando deja de cumplirse la precondición de X. Esto es así debido a que, cuando deja de cumplirse la precondición de X se eliminan todas las acciones del mismo plan que X, salvo las acciones internas como *terminaX*, que deben dejar el estado interno a su valor inicial.

La mayoría de las reacciones que lanzan una única acción abstracta siguen este modelo, como las reglas que lanzan *huir, cazar o eludirEnemigo*.

3.8. Ontología

El otro gran componente desarrollado en la nueva versión de COGNITIVA, tras la Interacción Social, es la implantación de una Ontología para gestionar el conocimiento. Este desarrollo surge a partir de dos necesidades muy definidas y claramente improrrogables: la excesiva lentitud en la velocidad de respuesta del sistema, y la necesaria adecuación de la gestión del conocimiento del agente a estándares globales.

El desarrollo de la ontología se ha basado en dos fases:

1. En la primera fase se diseñó una ontología que representara el conocimiento del agente con la herramienta Protégé, obteniendo un fichero en formato OWL con el esquema de la ontología.

2. En la segunda fase se sustituyó el sistema de conocimiento de COGNITIVA basado en listas de creencias e intereses por el sistema de ontología a través del framework de Java para ontologías, Jena. El acceso a los métodos de Jena se hizo a través de una API [García de la Torre, S.N.] que aislaba de conocer la arquitectura interna de Jena, permitiendo el manejo de objetos Entidad, Propiedad y Valor. Además fue necesario el desarrollo de una interfaz entre COGNITIVA y API antes mencionada para traducir el manejo de objetos Creencia o Interés a los objetos del tipo antes mencionado, la clase AccesoOntoCognitiva.

Todos los detalles adicionales acerca del desarrollo del sistema de ontología, su diseño e implementación se recogen en [Cabeza, S.N.]; donde se profundiza ampliamente en este componente.

Capítulo 4

INTERACCIÓN SOCIAL

4.1. Introducción

Los Sistemas multiagente que se pueden construir partiendo de una arquitectura de capas como COGNITIVA se basan en un conjunto de entidades autónomas o agentes, en los que cada agente no tiene información completa ni la capacidad para resolver el problema, es decir, tienen puntos de vista limitados. Además en estos sistemas no hay una entidad de control global, los datos están descentralizados y repartidos entre los agentes.

Más aún, el éxito del paradigma basado en agentes se basa en la existencia de entidades software heterogéneas y distribuidas que se comunican entre sí; no es fácil encontrar aplicaciones desarrolladas con orientación a agentes que se basen en un solo agente.

Sin embargo, utilizar varios agentes plantea serios desafíos, algunos de ellos de cierta complejidad. Estos planteamientos son la base del nivel social, propuesto en la arquitectura, pero, como se señaló en 2.10, no desarrollado. Algunos de estos planteamientos se enumeran a continuación, y son la base del trabajo expuesto en [Spinola, 2008]:

- Realizar una correcta descomposición del problema, es decir, planificación social.
- Garantizar la coherencia y la coordinación entre las actuaciones.
- Representar el conocimiento de otros agentes.
- Gestionar el uso de recursos.
- Desarrollar un método de comunicación entre agentes.

El presente trabajo soluciona el último punto, el desarrollo de un mecanismo de comunicación para los agentes basados en COGNITIVA.

La diversidad de agentes implica, en primer lugar, la necesidad de un entendimiento. Para ello se hace necesario tanto traducir el lenguaje de representación, como compartir el contenido semántico del lenguaje. Esto, cuando se habla de agentes, se traduce en un acuerdo en cuatro componentes:

1. Transporte, o cómo envían y reciben mensajes los agentes.
2. Lenguaje, para dotar de semántica a los mensajes.
3. Política, o la forma en que estructuran sus mensajes los agentes.
4. Arquitectura, para posibilitar la conexión entre sistemas.

4.2. Comunicación entre Agentes

4.2.1. Teoría de la Comunicación

Se define *Comunicar [DRAE, 2001]* como:

1. Hacer a otro partícipe de lo que uno tiene.
2. Descubrir, manifestar, hacer saber a alguno una cosa.
3. Conversar, tratar con alguno de palabra o por escrito.
4. Transmitir señales mediante un código común al emisor y al receptor.
5. Consultar, conferir con otros un asunto, tomando su parecer.

Todas estas definiciones hacen referencia a la comunicación en el sentido de comunicación *humana*, que es el modelo de referencia al que trata de aproximarse la comunicación entre agentes. La comunicación humana se compone de una serie de elementos o factores detallados en la figura 4.2.1, que muestra el modelo de Shannon y Weaver [Galeano, 1989] y que detallaremos a continuación:

- **Fuente**: Es el lugar de donde emana la información, los datos, el contenido que se enviará... en definitiva: de donde nace el mensaje primario.

- **Emisor o codificador**: Es el punto (persona, organización...) que elige y selecciona los signos adecuados para transmitir su mensaje; es decir, los codifica para poder llevarlo de manera entendible al receptor. En el emisor se inicia el proceso comunicativo.

- **Receptor o decodificador**: Es el punto (persona, organización...) al que se destina el mensaje, realiza un proceso inverso al del emisor ya que en él está el descifrar e interpretar lo que el emisor quiere dar a conocer. Existen dos tipos de receptor, el pasivo que es el que sólo recibe el mensaje, y el receptor activo o perceptor ya que es la persona que no sólo recibe el mensaje sino que lo percibe y lo almacena. El mensaje es recibido tal como el emisor quiso decir, en este tipo de receptor se realiza lo que comúnmente denominamos el feed-back o retroalimentación.

- **Código**: Es el conjunto de reglas propias de cada sistema de signos y símbolos que el emisor utilizará para trasmitir su mensaje, para combinarlos de manera arbitraria porque tiene que estar de una manera adecuada para que el receptor pueda captarlo. Un ejemplo claro es el código que utilizan los marinos para poder comunicarse; la gramática de algún idioma; los algoritmos en la informática..., todo lo que nos rodea son códigos.

- **Mensaje**: Es el contenido de la información (contenido enviado): el conjunto de ideas y acontecimientos expresados por el emisor y que desea trasmitir al receptor

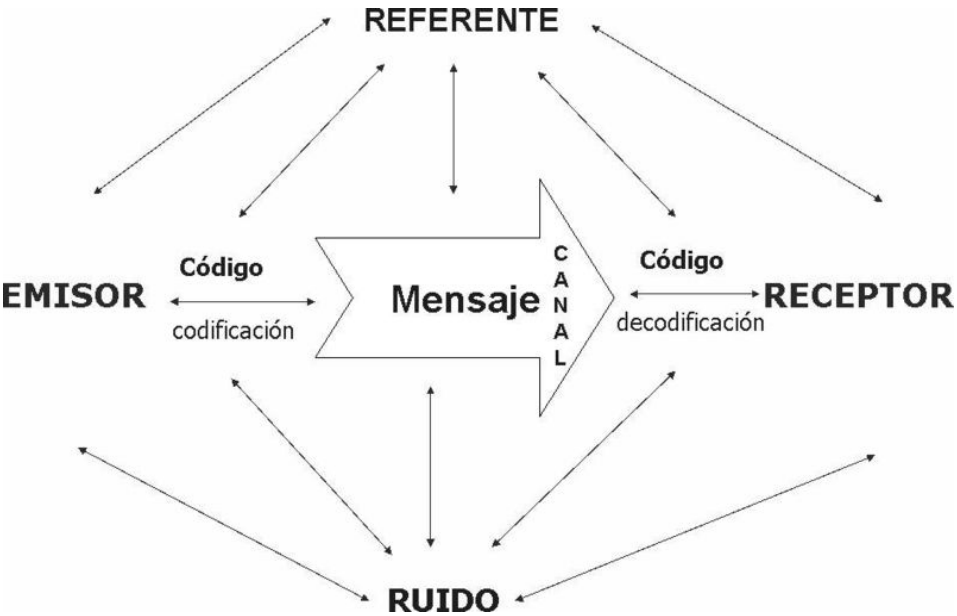

Figura 4.2.1: Elementos de la Comunicación Humana

para que sean captados de la manera que desea el emisor. El mensaje es la información.

- **Canal**: Es el medio a través del cual se transmite la información-comunicación, estableciendo una conexión entre el emisor y el receptor. Mejor conocido como el soporte material o espacial por el que circula el mensaje. Ejemplos: el aire, en el caso de la voz; el hilo telefónico, en el caso de una conversación telefónica.

- **Referente**: Realidad que es percibida gracias al mensaje. Comprende todo aquello que es descrito por el mensaje.

- **Situación**: Es el tiempo y el lugar en que se realiza el acto comunicativo.

- **Interferencia o barrera**: Cualquier perturbación que sufre la señal en el proceso comunicativo, se puede dar en cualquiera de sus elementos. Son las distorsiones del sonido en la conversación, o la distorsión de la imagen de la televisión, la alteración de la escritura en un viaje, la afonía del hablante, la sordera del oyente, la ortografía defectuosa, la distracción del receptor, ... También suele llamarse **ruido**.

- **Retroalimentación o realimentación (mensaje de retorno)**: Es la condición necesaria para la interactividad del proceso comunicativo, siempre y cuando se reciba una respuesta (actitud, conducta...) sea deseada o no, logrando la interacción entre el emisor y el receptor. Puede ser positiva (cuando fomenta la comunicación) o negativa (cuando se busca cambiar el tema o terminar la comunicación). Si no hay realimentación, entonces sólo hay información, pero no comunicación.

En el modelo de comunicación planteado para los agentes, será necesario trabajar en tres niveles, cada uno de los cuales relacionado con alguno de los componentes anteriormente explicados:

1. El nivel inferior es el método de interconexión entre los agentes, y se corresponde con el *canal* en el modelo de comunicación humana.

2. El nivel medio es el formato, o sintaxis, de la información. Su equivalente en el modelo humano es el *código.*

3. El nivel superior es el significado, o semántica, de la información; es la información en esencia y se corresponde con el *mensaje.*

En el presente trabajo nos centraremos en plantear un modelo centrado en el nivel superior, o semántica; por tanto se utilizarán tanto las arquitecturas como los lenguajes de comunicación entre agentes existentes, para definir los niveles inferior y medio respectivamente.

4.2.2. Mecanismos de transporte

Los mecanismos de transporte que pueden ser utilizados para lograr comunicación entre los agentes, tienen que cumplir una serie de requisitos fijados por las necesidades de los mensajes. A continuación se plantean algunas de las características que se exigen a una tecnología de comunicación entre agentes:

- Los mensajes deben poder ser planificables o servidos por eventos; es decir surgen tanto de procesos deliberativos como reactivos.
- La comunicación debe poder ser tanto síncrona como asíncrona
- El direccionamiento que los agentes debe poder hacerse tanto por direcciones físicas como por rol que realiza el agente.
- Los modos de envío deben soportar tanto unidifusión (unicast), multidifusión (multicast), como difusión (broadcast).

La implementación elegida, que cumple estos requerimientos, es el entorno JADE (Jade Agente DEvelopmente Framework) basado en el lenguaje de programación JAVA. El mecanismo de transporte de JADE es bastante flexible debido a que, de forma transparente, elige el mejor protocolo de transporte según la situación: Java RMI, notificación de eventos, HTTP e IIOP. De esta manera se consiguen cubrir todas las funcionalidades descritas anteriormente, aplicando el protocolo correcto para ofrecerla.

El más importante de ellos es el RMI (Remote Method Invocation) que habilita al programador para crear aplicaciones basadas en tecnología Java distribuida, en la cual los métodos de objetos en Java remotos pueden ser invocados desde otras máquinas virtuales de Java, posiblemente en diferentes servidores. El programa puede hacer una llamada sobre un objeto remoto, una vez que éste obtiene una referencia al objeto remoto, llamando a un servicio proveído por RMI o recibiendo la referencia como un argumento o un valor devuelto. RMI utiliza la serialización de objetos con parámetros oficiales y no oficiales, y no trunca los tipos, soportando un verdadero polimorfismo orientado a objetos.

4.2.3. Lenguajes de Comunicación

4.2.3.1. Fundamentos de los Lenguajes de Comunicación

La comunicación entre agentes (en la mayoría de los modelos) está basada en las teorías de los actos del habla ("Speech Acts Theories", [Austin, 1962]). Estas teorías están focalizadas en pragmática del lenguaje, es decir, son teorías del uso del lenguaje: intentan explicar cómo el lenguaje es usado por la gente para alcanzar sus metas e intenciones.

Se basan en las declaraciones formuladas con el lenguaje que provocan, de la misma manera que las acciones físicas, cambios en el estado del mundo. Por ejemplo: *Se abre la sesión, Declaración de guerra, ...* En general, todo lo que declaramos es expresado con la intención de satisfacer una meta, y las teorías de los actos del habla tratan de establecer como las declaraciones son usadas para alcanzar tales metas.

Para aclarar el concepto de acto del habla veremos cómo a partir del mismo contenido podemos elaborar diferentes actos del habla cambiando tan sólo la *performativa*, concepto que definiremos más adelante:

- Contenido : "La puerta está cerrada"
    - Request -> "por favor, cierra la puerta"
    - Inform -> "¡La puerta está cerrada!"
    - Inquire -> "¿Está cerrada la puerta?"

De esta manera observamos cómo los actos del habla no son meros predicados, sobre los que se pueda establecer una función de verdad (verdadero o falso); son acciones. Por tanto el acto del habla se define con tres aspectos diferenciados:

- Locución, el contenido en forma de predicado sujeto a verdadero o falso.
- Ilocución, la intención aplicada al contenido. Pueden ser de diversos tipos: Asertivas (*informar),* Directivas *(pedir y preguntar),* Comisivas *(prometer),* Declarativas *(causan eventos)*, Expresivas *(emociones y evaluaciones).*
- Perlocución, el efecto buscado con el acto, la acción que desea que se ejecute, aunque siempre fuera del control del agente.

El concepto antes mencionado de *performativa* no es más que la ilocución de una clase de expresiones, por ejemplo: *promise, report, tell, request, demand...*

Los lenguajes para la comunicación entre agentes son conocidos como ACL's (Agent Communication Lenguages). De entre ellos hay dos que son los más extendidos, el KQML/KIF y el FIPA ACL.

4.2.3.2. KQML/KIF

El lenguaje KQML/KIF (Knowledge Query and Manipulation Language/Knowledge Interchange Format) es un lenguaje y protocolo para intercambio de información y conocimiento. Los mensajes (performativas, como las antes definidas) son como cadenas ASCII siguiendo notación polaca-inversa de common LISP. Este lenguaje define tres componentes: un vocabulario, un lenguaje "interno" (KIF) y un lenguaje "externo" (KQML). De esta manera, un mensaje es una expresión *KQML*, en la que los argumentos son términos o sentencias *KIF*, formadas por palabras del vocabulario definido.

El vocabulario en KQML/KIF, se define por medio de una Ontología, que especifica una conceptualización del entorno de comunicación. Seguidamente, se acuerda un vocabulario común respecto al modelo definido en la Ontología.

El segundo componente, el lenguaje interno KIF, se basa en un lenguaje de predicados de primer orden. Utiliza la notación prefija (LISP) con extensiones para trabajar con lógicas no monótonas y definiciones complejas.

Por último, los mensajes del lenguaje externo KQML no sólo comunican frases en algún lenguaje sino que más bien comunican una actitud sobre el contenido. En síntesis, añaden intención al predicado de primer orden definido en KIF, por medio de las performativas. Su estructura es:

(<performativa> { :< palabras clave> <parámetros> })

Un ejemplo de mensaje en lenguaje KQML/KIF puede ser:

```
(ask
:sender Juan
:content (PRECIO IPOD ?precio)
```

```
:receiver servidor de stock
:reply-with IPOD-stock
:lenguage LISP
:ontology ALMACÉN )
```

4.2.3.3. FIPA ACL

El lenguaje FIPA ACL, fue desarrollado por FIPA (Foundation for Intelligent Physical Agents), organismo cuyo objetivo es lograr la estandarización en agentes desde 1996. Desde 2005, el estándar de FIPA está recogido dentro del corpus de estándares de IEEE. FIPA ACL define las pautas para que un lenguaje de comunicación sea FIPA-compatible, es decir, se adhiera al estándar propuesto por FIPA para el lenguaje de comunicación entre agentes. Los mensajes contienen uno o más parámetros, de los que sólo uno es obligatorio, la performativa, aunque es común que contengan también destinatario, remitente y parámetros de contenido.

La estructura de un mensaje FIPA ACL es similar a la de un mensaje KQML/KIF, debido a que ambos utilizan el mecanismo de performativa para denotar la intención del agente. Las diferencias entre un entorno y otro son realmente detalles del entorno semántico, como diferencias en la forma de registrarse o la gama de primitivas.

En KQML/KIF todo tipo de intercambio de bits de información entre los agentes se realiza a través de performativas, con primitivas para “registrarse”, “avisar”, “redirigir”, ... Sin embargo en FIPA ACL este tipo de intercambios, no orientados a la información en sí, son realizados mediante servicios proporcionados por los agentes. Otro tipo de servicios para los que KQML define primitivas y FIPA ACL no, son añadir o borrar información de las creencias de los agentes, o la capacidad de expresar objetivos y metas.

Además FIPA ACL utiliza un lenguaje diferente para expresar el contenido del mensaje, el lenguaje SL (Semantic Language), basado en lógica modal y legible para los humanos, ya que son cadenas de texto plano. Puede representar acciones, proposiciones y objetos. Un ejemplo de mensaje FIPA ACL:

```
(inform
:sender agente1
```

```
:receiver hpl-servidor-subastas
:content (precio (puja bien02) 150)
:in-reply-to ronda 4
:reply-with puja 04
:language sl
:ontology hpl-subasta
)
```

Para incorporar la comunicación entre agentes a COGNITIVA, se optó por utilizar el lenguaje de comunicación FIPA ACL al ser soportado por el entorno JADE y ser el más extendido, de forma que se aproxima a ser un estándar, objetivo de la comunidad FIPA.

Una vez escogido el lenguaje, se detallarán las capacidades del mismo. La principal capacidad de un lenguaje de comunicación entre agentes, una vez definido el lenguaje de contenido (SL), es su nivel de expresividad para comunicar las intenciones o *locuciones* del agente emisor. A continuación detallaremos las diferentes performativas de FIPA ACL que cumplen sobradamente los requerimientos de comunicación de COGNITIVA.

- accept-proposal: La acción de aceptar una propuesta recibida previamente para llevar a cabo una acción.
- agree: La acción de estar de acuerdo a llevar a cabo alguna acción, posiblemente en el futuro.
- cancel: La acción de cancelar alguna acción solicitada previamente con extensión temporal (es decir, que no es inmediata).
- call-for-proposal (cfp): La acción de pedir propuestas para llevar a cabo una acción.
- confirm: El emisor informa al receptor de que una proposición dada es cierta, donde es sabido que el receptor desconoce la certeza sobre la proposición.
- disconfirm: El emisor informa al receptor de que una proposición dada es falsa, donde es sabido que el receptor cree que la proposición es cierta.
- failure: La acción de contar a otro agente que una acción fue intentada, pero el intento falló.

- inform: El emisor informa al receptor de que una proposición es cierta.
- inform-if: Una macro acción del agente de la acción para informar al receptor si una proposición es o no verdadera.
- inform-ref: Una macro acción del emisor para informar al receptor de que un objeto corresponde a un descriptor definido.
- not-understood: El emisor del acto informa al receptor de que ha percibido que este último ha llevado a cabo una acción, pero no ha entendido que ha hecho exactamente.
- propose: La acción de enviar una proposición para llevar a cabo una cierta acción, dadas ciertas precondiciones.
- query-if: La acción de preguntar a otro agente si una proposición es cierta o no.
- query-ref: La acción de preguntar a otro agente sobre un objeto referido con respecto a una expresión.
- refuse: La acción de rechazar llevar a cabo una acción, explicando las razones del rechazo.
- reject-proposal: La acción de rechazar una propuesta para llevar a cabo alguna acción durante una negociación.
- request: El emisor solicita al receptor llevar a cabo alguna acción. Un tipo importante de usos de la performativa request es para solicitar al receptor llevar a cabo otro acto comunicativo.
- request-when: El emisor quiere que el receptor lleve a cabo alguna acción cuando ciertas proposiciones dadas sean verdad.
- request-whenever: El emisor quiere que el receptor lleve a cabo alguna acción en cuanto algunas proposiciones sean ciertas y cada vez que vuelvan a ser ciertas de nuevo.
- request-whomever: El emisor quiere que una acción sea llevada a cabo por algún agente distinto de sí mismo. El agente receptor debería, o intentar llevar a cabo la acción, o pasar la acción a algún otro agente.

- subscribe: El acto de solicitar una intención persistente de notificar al emisor acerca del valor de una referencia, y de notificarle otra vez cada vez que el objeto identificado por la referencia cambie.

Una vez escogido el lenguaje de comunicación, pasaremos a la siguiente subsección donde definiremos la semántica de la comunicación, es decir, los protocolos de intercambio de mensajes entre los agentes; tanto los propuestos por FIPA como los desarrollados e incorporados a COGNITIVA.

4.2.4. Protocolos de Comunicación

Los protocolos de comunicación representan los patrones que modelan las posibles comunicaciones. Para poder ser definido como tal, todos los participantes en la conversación deben conocerlo previamente, y éste debe de estar definido formalmente.

Cuando nos referimos a una instancia concreta de este patrón de intercambio de mensajes, hablamos de *conversación.* En una conversación, todos los agentes intervinientes deben conocer en qué protocolo se encuentran, y en qué fase del mismo. El patrón mínimo de intercambio es aquél en el que un agente envía y recibe información, de forma pasiva. Progresivamente, el agente, además de enviar y recibir información, puede ser capaz de pedirla. Esta solicitud puede ser además activa, o deliberativa. Finalmente, en modelos más complejos, el agente acumula recursos, información, genera planes cooperativos y establece negociaciones con otros agentes.

FIPA define un conjunto de protocolos que buscan la estandarización de los patrones de intercambio de mensajes más comunes, con el objetivo de cubrir las necesidades de comunicación típicas de los sistemas multi agente. En los siguientes apartados veremos algunos de estos protocolos. Su notación se basa en los *Agent Interaction Protocols* de AUML (ver [Odell et al., 2001]), una ampliación de UML para desarrollo orientado a agentes, que se utiliza los diagramas de secuencia para representar protocolos.

El último de la serie protocolos que se muestran a continuación, es el desarrollado en este proyecto a partir del protocolo Request estándar, y utilizado para gestionar las solicitudes de acciones de un agente a otro. Este protocolo sienta las bases del nivel social al definir un intercambio de mensajes mediante el cuál, un agente solicita a otro la realización de una o más acciones de su plan, de cara a satisfacer una meta.

4.2.4.1. Request

Este protocolo permite a un agente solicitar a otro que realice una acción. El receptor debe realizarla o responder que no puede. El intercambio de mensajes se puede observar en la figura 4.2.2.

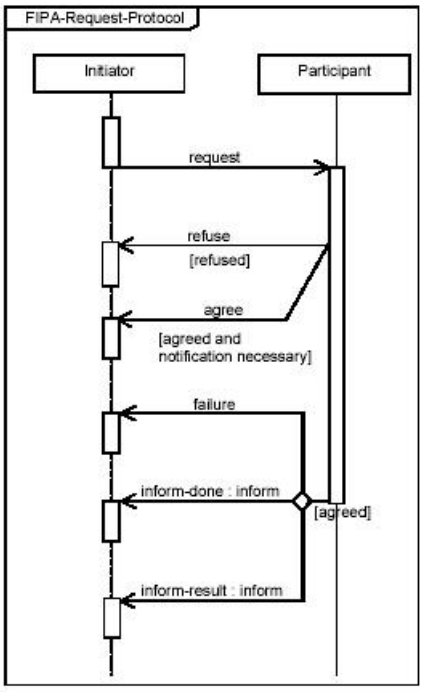

Figura 4.2.2: Protocolo FIPA Request

4.2.4.2. Query

Se emplea para solicitar a un agente una cierta información. Hay dos tipos, el query-if, para informaciones de tipo verdadero o falso, y el query-ref, para el resto. El intercambio de mensajes se puede observar en la figura 4.2.3.

4.2.4.3. Request When

Análogo al Request, pero en este caso el receptor debe de esperar a que se cumpla una precondición para responder. El intercambio de mensajes se puede observar en la figura 4.2.4.

4.2.4.4. Contract Net

Este protocolo se utiliza para hacer negociaciones en la contratación de una tarea. Un agente desea que se realice una acción, y hay varios candidatos dispuestos. El objetivo es llegar a un acuerdo con el agente que minimice una función que caracterice a dicha tarea, por ejemplo, el precio.

Inicialmente, el manager, con el rol FIPA de iniciador, genera *m* mensajes del tipo cfp (*call for proposal)* y queda a la espera durante un determinado tiempo, después del cual no recibirá más mensajes (un total de *n* recibidos).

Sea *i* el número de mensajes de tipo refuse, tendremos entonces $j = n - i$ mensajes del tipo propose. Para cada uno de los *j* mensajes, se envía posteriormente un accept-proposal o un reject-proposal. Se informa del resultado con un failure o con un inform, acompañado o no del resultado. El intercambio de mensajes se puede observar en la figura 4.2.5.

4.2.4.5. Iterated Contract Net

Es una ligera variación del Contract Net que permite el número de rondas que sea necesario. Tiene el propósito para el iniciador de la interacción de conseguir mejores ofertas

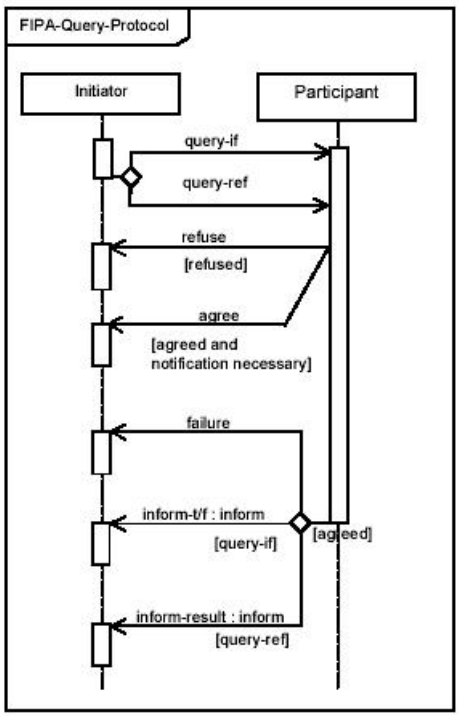

Figura 4.2.3: Protocolo FIPA Query

de los interlocutores mediante argumentación o crítica de las ofertas anteriores.

Inicialmente, el agente con el rol de iniciador de la conversación genera *m* mensajes cfp. Después de un deadline de espera, el iniciador recoge *n* ofertas. Sean un total de *j* las que rechazan realizar la tarea mediante refuse. Tenemos entonces $k = n - j$ ofertas de agentes que están dispuestos a realizar la tarea con propose. Si la iteración no es la última, de entre el total de *k* ofertas recibidas, se rechazarán algunas directamente, $k - I$, y se aceptarán otras tantas *I*. De entre las aceptadas, se elabora una contra oferta para cada agente, y se envuelve en un nuevo cfp.

El intercambio de mensajes se puede observar en la figura 4.2.6.

4.2.4.6. Brokering

Tiene como propósito permitir interaccionar con otros agentes a través de un mediador (el broker). Proxy es una macro (incluye otro acto comunicativo que el broker debe hacer llegar al seleccionado o seleccionados). El broker devuelve los resultados mediante

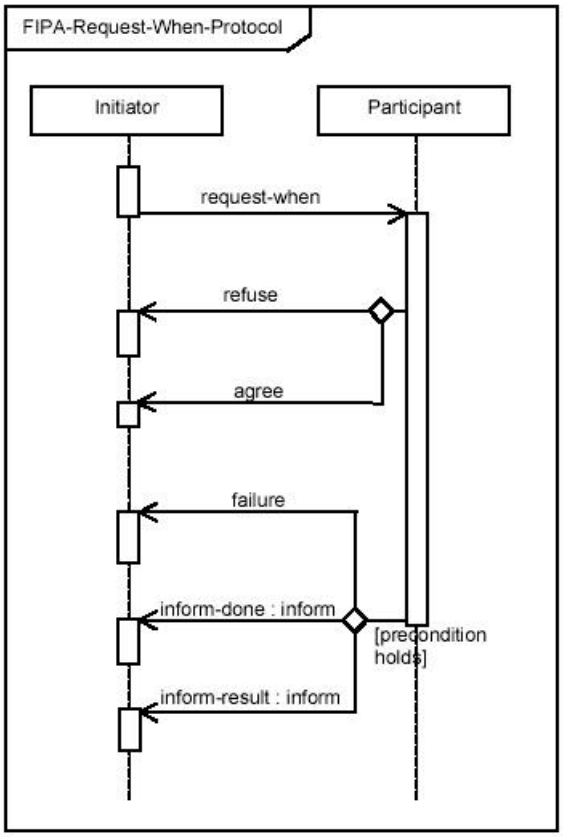

Figura 4.2.4: Protocolo FIPA Request When

un reply-message-sub-protocol (por ejemplo, un reply con la respuesta en el cuerpo del mensaje).

El intercambio de mensajes se puede observar en la figura 4.2.7.

4.2.4.7. Recruiting

Tiene como propósito reclutar agentes para interaccionar con ellos posteriormente. Es usado en entornos basados en mediadores, y se diferencia del anterior en que ya no es el broker el que interactúa con los seleccionados, sino el receptor designado. El intercambio de mensajes se puede observar en la figura 4.2.8.

4.2.4.8. Subscribe

De utilidad para subscripción a un servicio de notificación de eventos. El agente participante se compromete (mediante la emisión de un agree) a informar mediante inform cada vez que el evento se produzca. El intercambio de mensajes se puede observar en la figura 4.2.9.

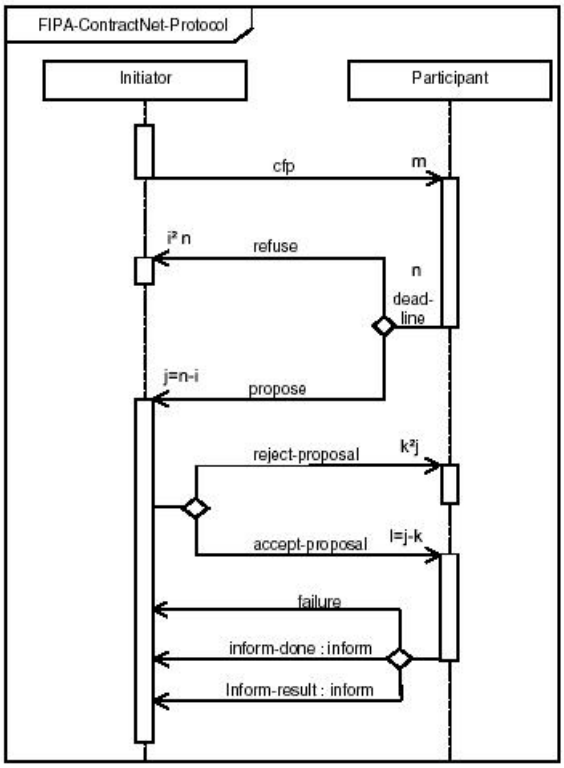

Figura 4.2.5: Protocolo FIPA Contract Net

4.2.4.9. Propose

El agente iniciador propone al otro agente participante, que lo ha de supervisar, el realizar una acción. El intercambio de mensajes se puede observar en la figura 4.2.10.

4.2.4.10. English Auction

El iniciador ofrece un precio. Si ningún posible comprador realiza una oferta, se vuelve a ofrecer otro precio. Se selecciona el mejor de entre los participantes que emiten un propose. El intercambio de mensajes se puede observar en la figura 4.2.11.

4.2.4.11. Solicitar Servicio

Este protocolo 4.2.12 es muy similar al Request estándar definido por FIPA. Sin embargo, en la siguiente sección veremos como todos los mensajes están encapsulados dentro de

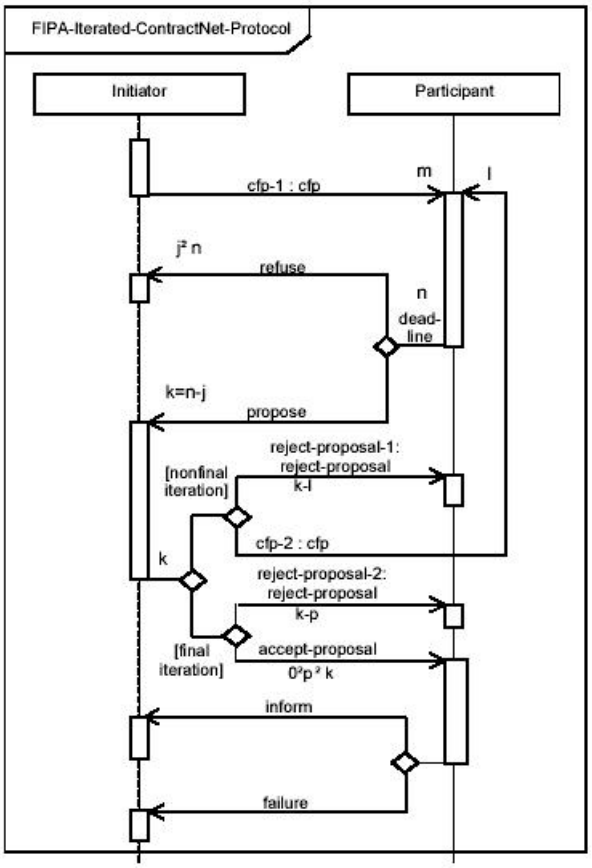

Figura 4.2.6: Protocolo FIPA Iterated Contract Net

acciones (en el sentido COGNITIVA) y éstas a su vez en otras de tipo abstracto; de forma que el protocolo cobra un sentido más completo al incluir también el procesamiento y la lógica de los agentes respecto a los mensajes recibidos.

4.3. Diseño

En esta sección se tratará acerca de la forma en que se han implantado los mecanismos de Interacción Social en COGNITIVA.

4.3.1. Acciones

El objetivo prioritario a la hora de abordar el diseño de estos mecanismos fue aprovechar al máximo todo lo ya definido en COGNITIVA, objetivo que fue abordable gracias a la gran capacidad de la arquitectura para ampliarse con nuevas capacidades.

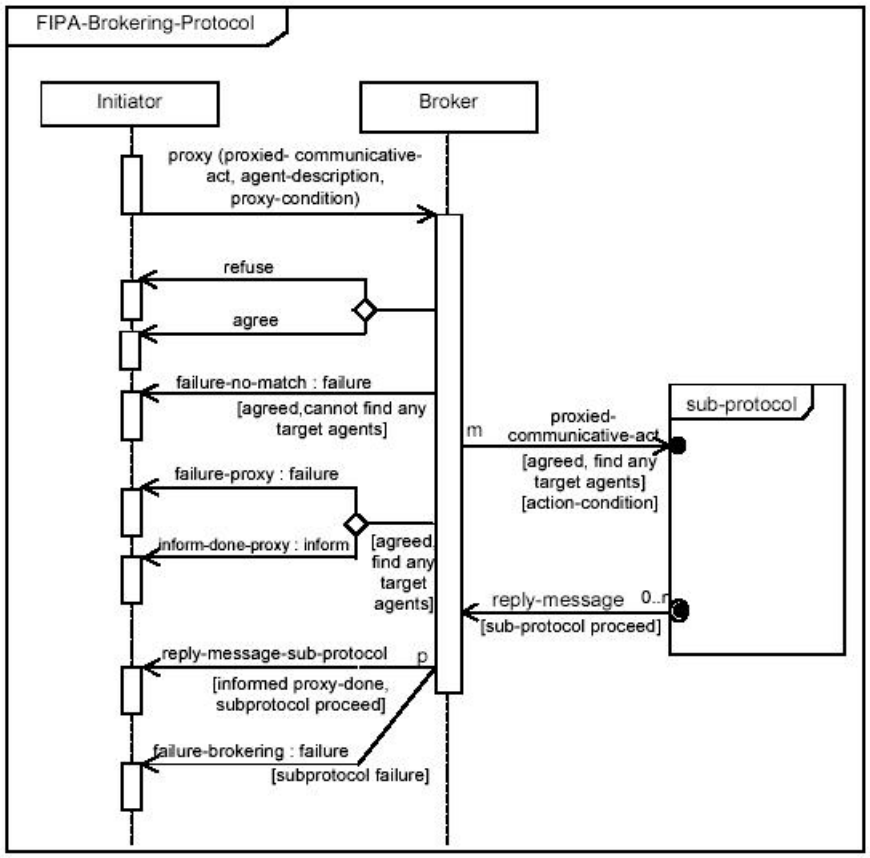

Figura 4.2.7: Protocolo FIPA Brokering

Concretamente, se optó por encapsular los mensajes FIPA emitidos entre los agentes dentro de acciones COGNITIVA, del tipo Acciones Externas. Esto permite adaptar el estándar de mensajes FIPA al formato de acciones de COGNITIVA, de manera que ambos conviven de forma transparente al programador gracias a los acuerdos establecidos acerca de los parámetros del mensaje.

Se ha conseguido que, con mínimas modificaciones en los actuadores, el mismo formato de Acción sirva tanto para efectuar una acción sobre el entorno, como se hacía hasta ahora, como para poder comunicarse con otro agente. Para conseguir distinguir entre las acciones efectuadas sobre el entorno (Técnicas) y las de interacción con otro agente (Sociales) se ha establecido que las acciones de tipo Acción Externa pasen a tener una serie de parámetros fijos, determinados al estar en las primeras posiciones de la lista de parámetros de la acción, en función de los cuales la acción es de un tipo u otro:

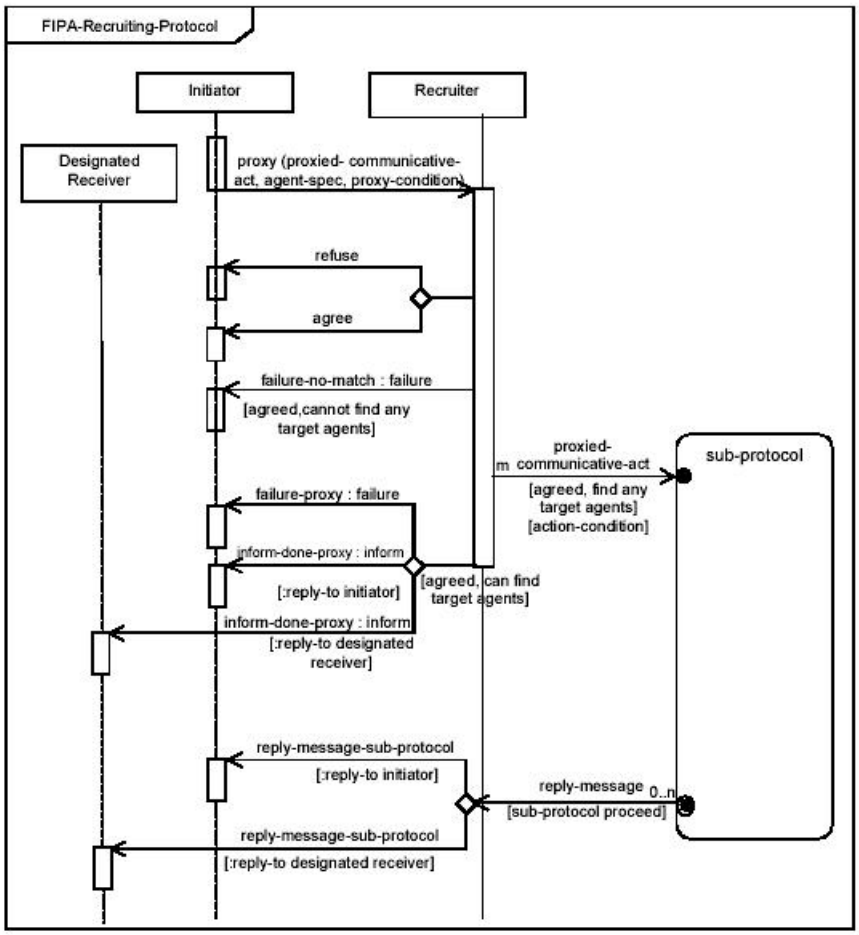

Figura 4.2.8: Protocolo FIPA Recruiting

Valores de los Parámetros de las Acciones Externas Técnicas

1 Nombre Nombre de la Acción

2 Destino *Actuador*

3 Performativa *INFORM*

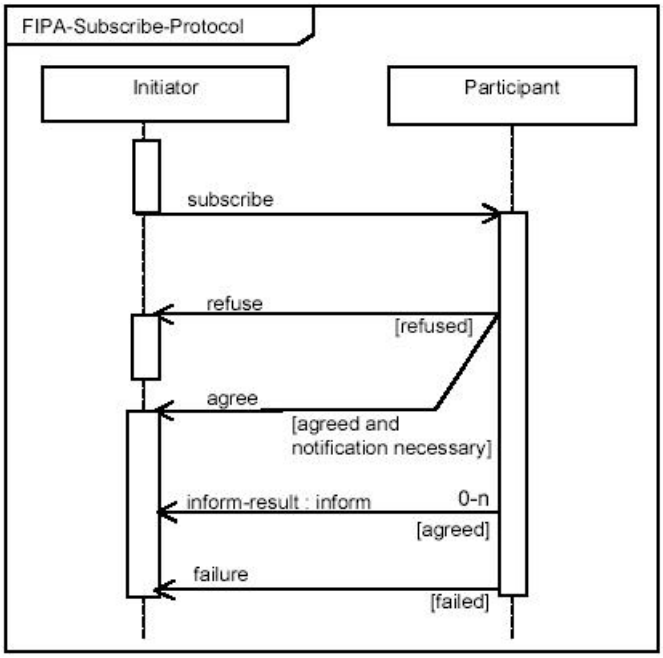

Figura 4.2.9: Protocolo FIPA Subscribe

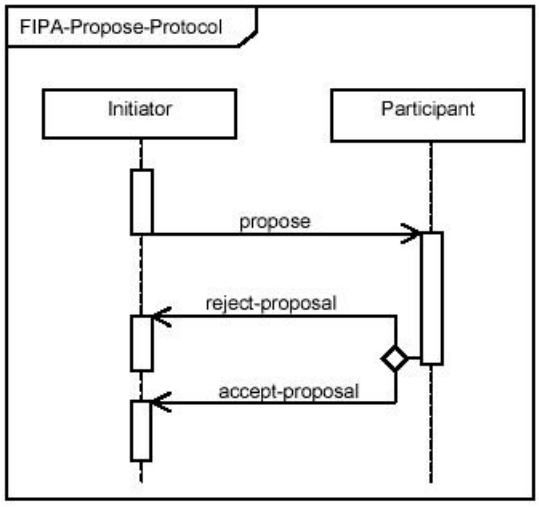

Figura 4.2.10: Protocolo FIPA Propose

4 Protocolo *AgenteActua*

[5…N] Parámetros particulares de la Acción

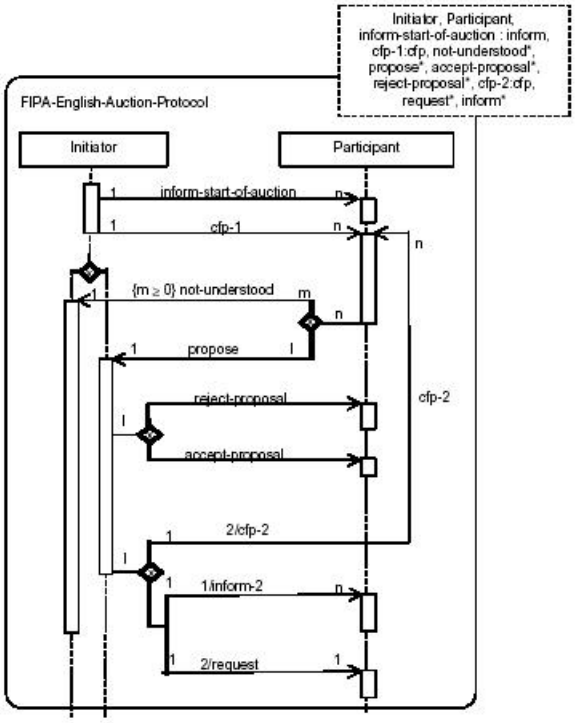

Figura 4.2.11: Protocolo FIPA English Auction

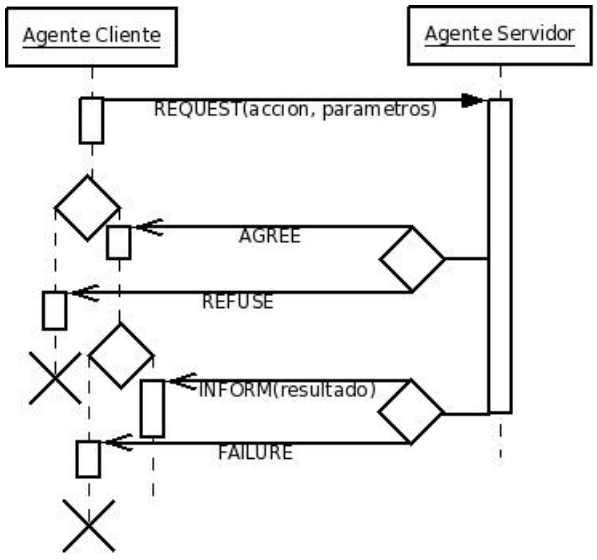

Figura 4.2.12: Protocolo Solicitar Servicio

Valores de los Parámetros de las Acciones Externas Sociales

1 Nombre Nombre de la Acción

2 Destino AID del Agente con el que comunicarse

3 Performativa Performativa que corresponda según la fase del protocolo

4 Protocolo *SolicitarServicio*

[5...N] Parámetros particulares de la Acción

De esta manera, se posibilita que las clases de la jerarquía de Acciones (ver 3.1.2) permanezcan inalterables, puesto que para añadir estos nuevos subtipos de acciones sólo será necesario rellenar los parámetros según proceda en el método *parametrosDinamicos* . A partir del parámetro número 5 se incluyen ahora los parámetros particulares de cada acción

Hay que señalar que el parámetro *Protocolo* tomaría nuevos valores al incluir nuevos protocolos de comunicación en la arquitectura, como pueden ser protocolos de negociación, o de incertidumbre; así se garantiza la escalabilidad del modelo.

Por otro lado, para encapsular los mensajes del protocolo de solicitud planteado en 4.2.4.11, se definen una serie de acciones que se declaran a continuación:

4.3.1.1. Acciones del Agente Cliente

solicitar:

Precondiciones:

- = (Tipo(Agente_Servidor), Servidor)
- < (Distancia(Agente_Servidor),10)

Operador:

- solicitar(accion, parámetros)

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 30 segundos.

Solicitar es una acción abstracta que da comienzo al protocolo desde el agente cliente. Tiene como parámetros la *acción* a solicitar y los *parámetros* de dicha acción. Se descompone en:

1. enviarSolicitud()
2. tratarRespuesta()
3. terminarSolicitud()

enviarSolicitud:

Precondiciones:

- Ninguna

Operador:

- mensaje(REQUEST, accion, parámetros)

Consecuencias:

- Ninguna

Expectativas:

- Acontecimiento: = (Estado_Comunicación (Agente_Servidor), esperando_servicio)

- Expectación: <Absolutamente>
- Deseo: <Absolutamente>

- Acontecimiento: = (Estado_Comunicación (Agente_Servidor), inicial)
  - Expectación: <Nada>
  - Deseo: <Nada>

Indicador de caducidad:

- 30 segundos.

Esta acción, del subtipo Acción Externa *Social* es la encargada de enviar el mensaje REQUEST al agente servidor.

En este punto se ha profundizar en el nuevo uso dado a las expectativas en este desarrollo de interacción social. Durante la interacción con otro agente, las expectativas respecto al estado en que se encuentre la comunicación con él, se utilizarán para mantener al agente cliente parado en espera de una respuesta del servidor. Mientras no llegue a su fin la caducidad asignada a la acción, el agente cliente permanece a la espera de que, o se satisfaga alguna de las expectativas, o se dé por imposible su cumplimiento, lo que, traducido al protocolo significa, o bien que el servidor responda y se reciba contestación (positiva o negativa), o que se asuma que no contestará y se abandone el protocolo.

Este mecanismo se ha revelado como la mejor manera de que el protocolo se lleve a cabo en un tiempo razonable (si todo va bien), o se deseche rápidamente (si algo va mal) dejando el estado de los agentes en perfecta coherencia.

tratarRespuesta:

Precondiciones:

- Ninguna

Operador:

- tratarRespuesta()

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 30 segundos.

Esta acción abstracta analiza la respuesta del agente cliente, y en función de ésta, se descompone continuando el protocolo, o terminándolo:

- Si = (Estado_Comunicación (Agente_Servidor), esperando_servicio) entonces:
  1. recibirResultado()
  2. tratarResultado()
- En otro caso entonces:
  1. cancelar()

recibirResultado:

Precondiciones:

- Ninguna

Operador:

- recibirResultado()

Consecuencias:

- Ninguna

Expectativas:

- Acontecimiento: = (Estado_Comunicación (Agente_Servidor), finalizando_solicitud)
  - Expectación: <Absolutamente>
  - Deseo: <Absolutamente>
- Acontecimiento: = (Estado_Comunicación (Agente_Servidor), inicial)
  - Expectación: <Nada>
  - Deseo: <Nada>

Indicador de caducidad:

- 30 segundos.

Esta acción cumple la función de mantener al agente cliente en espera del resultado del servicio por medio del mecanismo de expectativas, explicado en la acción *enviarSolicitud*. Dado que esta acción no envía operadores a otros agentes, podría pensarse que, conceptualmente, estamos ante una acción interna; sin embargo el uso de expectativas es contrario a la definición de acción interna (las acciones internas no tienen expectativas ya que el agente, por ser internas, conoce de antemano su resultado). Por tanto, dentro de las Acciones Externas de tipo social, hay que incluir las acciones que, como ésta, no envían mensaje, sino que lo reciben. Para ello se modifica el parámetro fijo *Destino:*

2. Destino *Nadie*

tratarResultado:

Precondiciones:

- Ninguna

Operador:

- tratarResultado()

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 30 segundos.

Esta acción abstracta analiza el resultado enviado por el agente cliente, y en función de éste, cancela el protocolo o lo finaliza

- Si = (Estado_Comunicación (Agente_Servidor), finalizando_solicitud) entonces:
    - No se descompone en nada, ya que, la acción *terminarSolicitud* que finaliza el protocolo la inserta la acción *Solicitar* por lo que esta ya se encuentra en el Organizador.
- En otro caso, entonces:
    1. cancelar()

cancelar:

Precondiciones:

- = (Estado_Interno (Agente_Cliente), cancelado)

Operador:

- cancelar()

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 5 segundos.

Esta acción se dispara para cancelar el protocolo en cualquiera de sus fases: tanto al recibir un REFUSE, como al recibir un FAILURE, o en algún otro momento en que el cliente decida cancelarlo. Es una acción interna, pero más bien podríamos definirla como meta-acción, ya que su funcionamiento es el siguiente: la precondición está definida de forma que nunca se cumpla, por tanto, cuando la acción llega al Organizador, ésta no se cumple y se cancela el plan. De esta manera la acción en realidad nunca llega a ejecutarse, al cancelar el plan se eliminan todas las acciones del mismo, dejando coherente el estado del agente.

terminarSolicitud:

Precondiciones:

- Ninguna

Operador:

- terminarSolicitud()

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 5 segundos.

La acción interna *terminarSolicitud* se encarga de dejar coherente el estado de los agentes tras finalizarse el protocolo. Se ejecuta tanto en el agente cliente como en el servidor, y de la misma manera cuando finaliza satisfactoriamente como cuando se cancela por algún motivo. Su finalidad es cambiar el Estado de la Comunicación con respecto al otro agente al estado *inicial.*

4.3.1.2. Acciones del Agente Servidor

tratarSolicitud:

Precondiciones:

- Ninguna

Operador:

- tratarSolicitud()

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 30 segundos.

Esta acción abstracta, recoge los parámetros de la solicitud enviada por el agente Cliente en la acción *enviarSolicitud*. Éstos han sido convertidos en creencias en el comportamiento *Social* del agente Servidor (ver 4.3.2). Se descompone en:

- Si la acción solicitada se encuentra dentro de la colección del agente Servidor:
  1. enviarConfirmacion()
  2. tratarServicio()
  3. terminarSolicitud()
- En otro caso, entonces:
  1. enviarRechazo()
  2. terminarSolicitud()

enviarConfirmacion:

Precondiciones:

- Ninguna

Operador:

- mensaje(AGREE)

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 6 segundos.

Esta acción encapsula el mensaje de confirmación destinado al agente Cliente, para indicarle que el agente Servidor tratará de llevar a cabo el servicio solicitado.

enviarRechazo:

Precondiciones:

- Ninguna

Operador:

- mensaje(REFUSE)

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 6 segundos.

Esta acción encapsula el mensaje de rechazo destinado al agente Cliente, para indicarle que el agente Servidor no llevará a cabo su servicio.

tratarServicio:

Precondiciones:

- Ninguna

Operador:

- tratarServicio()

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 30 segundos.

Esta acción abstracta se encarga de evaluar el resultado de la acción realizada, para ver si ésta se ha ejecutado con éxito, o ha ocurrido algún problema que haya impedido su correcta ejecución. Se descompone en:

- Si = (Estado_Comunicación (Agente_Cliente), fallando_servicio)
    1. enviarFallo()
- Si = (Estado_Comunicación (Agente_Cliente), realizando_servicio)
    1. enviarResultado()

enviarFallo:

Precondiciones:

- Ninguna

Operador:

- mensaje(FAILURE)

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 6 segundos.

Esta acción encapsula el mensaje de fallo destinado al agente Cliente, para indicarle que ha habido algún problema al tratar de realizar el servicio solicitado, y éste no se ha podido llevar a cabo de forma satisfactoria.

enviarResultado:

Precondiciones:

- Ninguna

Operador:

- mensaje(INFORM, resultado)

Consecuencias:

- Ninguna

Expectativas:

- Ninguna

Indicador de caducidad:

- 6 segundos.

Esta acción encapsula el mensaje de información de resultado destinado al agente Cliente, para indicarle que la solicitud requerida ha sido realizada satisfactoriamente y comunicar-

le los resultados de la ejecución (en el caso de que los hubiera).

4.3.2. Comportamiento Social

Junto con la definición del protocolo de Solicitud de Servicio, y de las Acciones que posibiliten la comunicación y encapsulen los mensajes, surge la necesidad de incorporar a los agentes un mecanismo que permita la recepción de los mensajes y actualice el Estado de la Comunicación en función de éstos.

En el conjunto de comportamientos de los agentes, encontramos el comportamiento *esperaCambio* que se encarga de recibir los mensajes de cambios en el entorno y/o los agentes, enviados por el Agente de Percepción. De igual manera, se trató de desarrollar un comportamiento que recibiera los mensajes de comunicación con el resto de agentes, y además mantener el estado de la conversación y actualizarlo con el contenido de dichos mensajes.

Así se desarrolla el comportamiento *social*, cuyo diagrama de comportamiento (ver definición en la sección 3.1.3) se puede observar en el cuadro 4.1. La misión este comportamiento es actualizar el Estado de la Comunicación con cada agente, permitiendo que se puedan mantener conversaciones con varios agentes a la vez.

Esta actualización se hace a partir de la performativa del mensaje recibido, junto con el estado actual de la conversación, de forma que se sigan los grafos de estados de la figuras 4.3.2 y 4.3.2. Otro tipo de cambios de estado pueden deberse a Acciones Externas de tipo social ejecutadas por el propio agente, es decir, a mensajes enviados por el propio agente.

Nombre del Comportamiento: social	
Agente	AgenteSimple
Protocolos	SolicitarServicio
action	Al recibir un mensaje del protocolo "SolicitarServicio": ▪ Si es un REQUEST, cambiamos el estado de la comunicación a 'analizando solicitud' y volcamos los parámetros del servicio a creencias ▪ Si es un AGREE, cambiamos el estado de la comunicación a 'esperando servicio' ▪ Si es un NOT UNDERSTOOD | REFUSE | FAILURE, cambiamos el estado de la comunicación a 'inicial' ▪ Si es un INFORM, cambiamos el estado de la comunicación a 'finalizando solicitud'
onStart	Nada
onEnd	Nada
done	Este comportamiento es cíclico (no termina nunca)

Tabla 4.1: Diseño del Comportamiento "social"

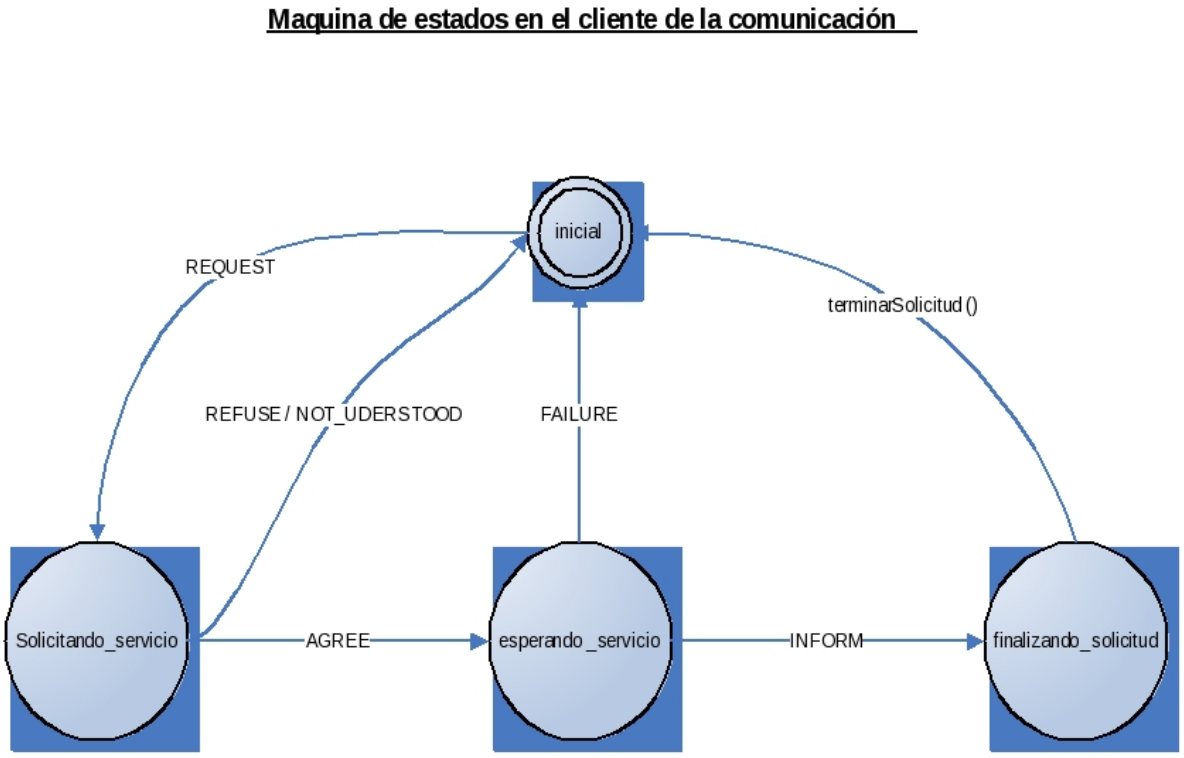

Figura 4.3.1: Estados del protocolo Solicitud Servicio (Agente Cliente)

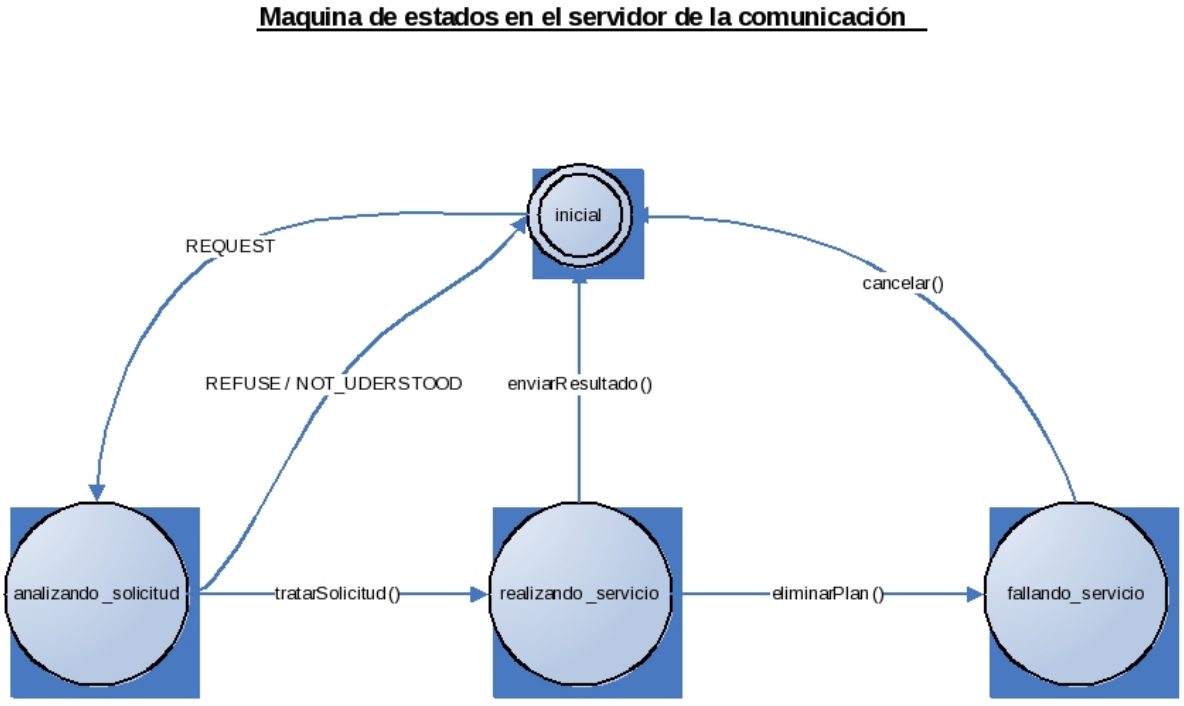

Figura 4.3.2: Estados del protocolo Solicitud Servicio (Agente Servidor)

Capítulo 5

CONCRECIÓN CONTEXTUAL GHOST SQUADRON

5.1. Introducción

El sistema para el que se aplicará la fase de Concreción Contextual, a partir de la Concreción Funcional extendida y mejorada explicada en el capítulo 3, es un EVI (Entorno Virtual Inteligente) con características lúdicas denominado *Ghost Squadron*.

El desarrollo de este EVI forma parte de una de las evaluaciones realizadas en [Aguilar, 2008], trabajo que trata sobre el estudio de Estrategias de Entrenamiento Colaborativo que utilicen EVI's. Las necesidades planteadas requerían un sistema multiagente que proporcionara capacidades tanto reactivas como sociales, por tanto fue utilizado para poner a prueba las nuevas funcionalidades de interacción social desarrolladas en 4.

El entorno es 2D, y en él se combinan agentes virtuales con humanos, de cara a evaluar el entrenamiento de grupos en los que uno o más de los participantes sean virtuales. Estos agentes virtuales serán denominados agentes PANCHO, utilizando la misma denominación empleada en [Aguilar, 2008], y que responde a las siglas inglesas para "Pedagogical AgeNt to support Collaborative Human grOups"

5.2. Planteamiento del Escenario

El escenario colaborativo ha sido diseñado como un aplicación lúdica, o videojuego, de tipo estratégico. Cada uno de los miembros tiene un conjunto de habilidades y responsa-

bilidades específicas; así mismo, el entorno genera información diferente para cada uno de ellos, de manera que cada uno tiene una visión parcial del escenario. De esta manera se fuerza a que el equipo mantenga una interacción permanente en la ejecución.

La interacción, desarrollada gracias al nuevo componente de Interacción Social, dependerá de la dificultad de la tarea, las habilidades colaborativas de cada miembro y de su personalidad.

5.2.1. Misión del grupo

El equipo debe trasladar por vía aérea a un diplomático desde una zona de seguridad a otra, atravesando una zona enemiga. En la zona enemiga, mantienen vigilancia constante aviones y buques enemigos, todos ellos en continuo movimiento. Lo prioritario será evitar entrar en conflicto, tratando en todo momento de no ser detectado mientras se sobre-vuela la zona enemiga.

5.2.2. Personajes en el grupo

El escuadrón está conformado por un grupo de tres personajes:

- *Piloto*: Responsable de dirigir el avión hacia la zona de destino, intentando realizar el trayecto por espacios aéreos seguros.
- *Oficial Aéreo:* Responsable de vigilar el espacio aéreo, para orientar al piloto hacia espacios seguros. En caso de una situación de riesgo, es el responsable de la acción ofensiva aérea.
- *Oficial Terrestre:* Responsable de vigilar el espacio terrestre o marítimo, para orientar al piloto hacia espacios seguros. En caso de una situación de riesgo, es el responsable de la acción ofensiva marítimo-terrestre.

Cada uno de los tres personajes posee habilidades diferenciadas vinculadas con el rol funcional del personaje.

Así mismo, cada uno de los tres recibe información parcial del escenario; el piloto no conoce de manera directa la posición de los vehículos enemigos, sin embargo, es el único que tiene conocimiento de la posición del avión con respecto al origen y destino del vuelo; por su parte, cada uno de los dos oficiales (aéreo y terrestre) mantiene información

específica de la ubicación (aérea o terrestre) de los vehículos enemigos, y por tanto, del estado de seguridad de la nave.

5.2.3. Dinámica de la Actividad

En el trayecto se requiere atravesar una zona enemiga que mantiene vigilancia aérea por encima de los 5.000 metros, así como marítima por debajo de dicha altitud.

La estrategia de vigilancia el enemigo consiste en mantener tres buques que cubren el 60 % de la zona vigilada (ver figura 5.2.1) informando en todo momento de una posible intromisión en el espacio marítimo, así como en el aéreo (por debajo de los 5.000 metros). Los buques enemigos pueden emprender una ofensiva sobre el espacio en el que mantienen vigilancia.

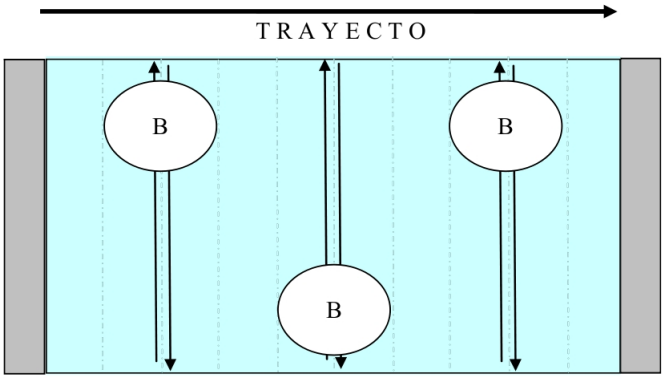

Figura 5.2.1: Zona de vigilancia por vía marítima

Además, un conjunto de aviones enemigos sobrevuela toda la zona cada cierto tiempo (ver figura 5.2.2), informando del estado encontrado de la zona. La aeronave en la cual se desplaza el escuadrón puede volar a una velocidad máxima, que rebasa (en un 20 %) la rapidez de los aviones de vigilancia, y alcanza una altitud similar a la de dichos aviones enemigos (hasta 12.000 metros).

En caso de haber detectado algún intruso, el avión enemigo puede emprender una ofensiva sobre el escuadrón.

El vehículo en el que se desplaza el escuadrón cuenta con un radar con un alcance que rebasa en un 20 % el de los vehículos enemigos, lo que le permite identificar, sin ser percibido, a aviones o buques enemigos con un pequeño margen de antelación. Dicho conocimiento le será de utilidad en la toma de decisiones en grupo, ya sea para reconducir el trayecto, modificar la altitud, o iniciar una ofensiva.

El éxito de la misión dependerá de la colaboración y astucia de los integrantes del equipo, sobre todo en aquellas decisiones y acciones grupales que le permitan al equipo librar situaciones de riesgo, y cumplir con su misión.

La interfaz de los Mandos Terrestre y Aéreo para dar órdenes al Piloto y recibir información del entorno, es la mostrada en las figuras5.2.3 y 5.2.4.

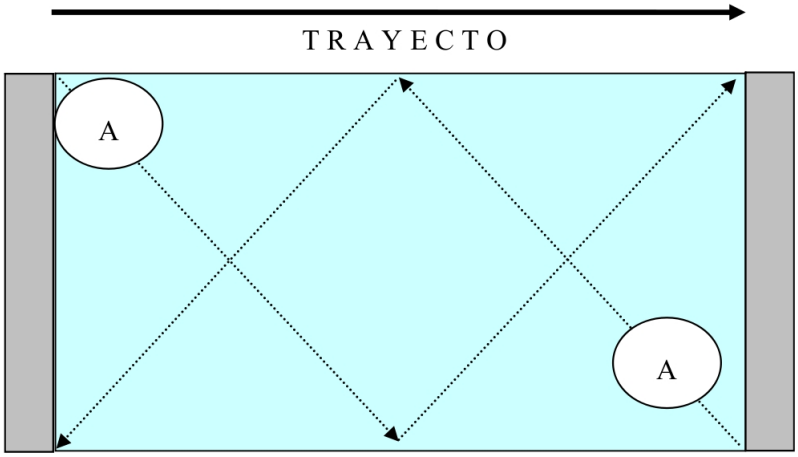

Figura 5.2.2: Zona de vigilancia por vía aérea

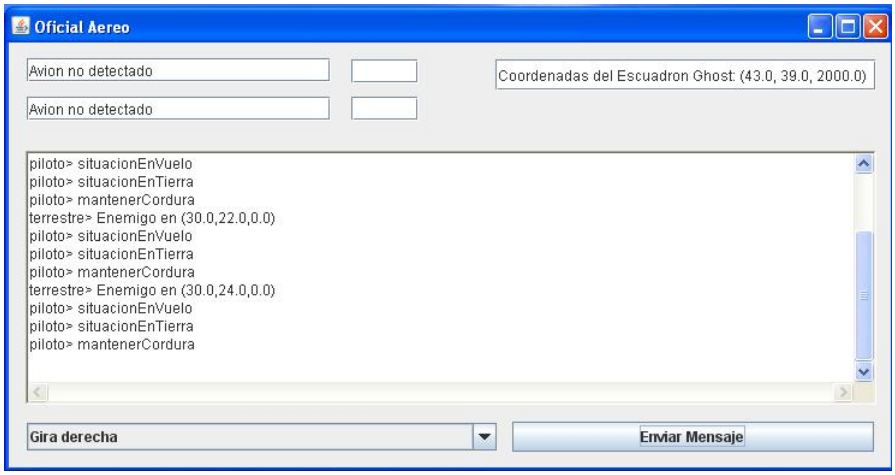

Figura 5.2.3: Interfaz Mando Aéreo *Ghost Squadron*

El mando aéreo recibe información de las coordenadas de los aviones enemigos, siempre que éstos se encuentren dentro de su campo de visión, en el flanco superior izquierdo de su pantalla. A su vez, recibe permanentemente información acerca de la posición del piloto en el flanco superior derecho de la pantalla.

En la parte central mantiene un registro con los mensajes intercambiados entre ambos mandos y el piloto, y finalmente, en la franja inferior posee un selector con los posibles mensajes a emitir y un botón para enviarlos.

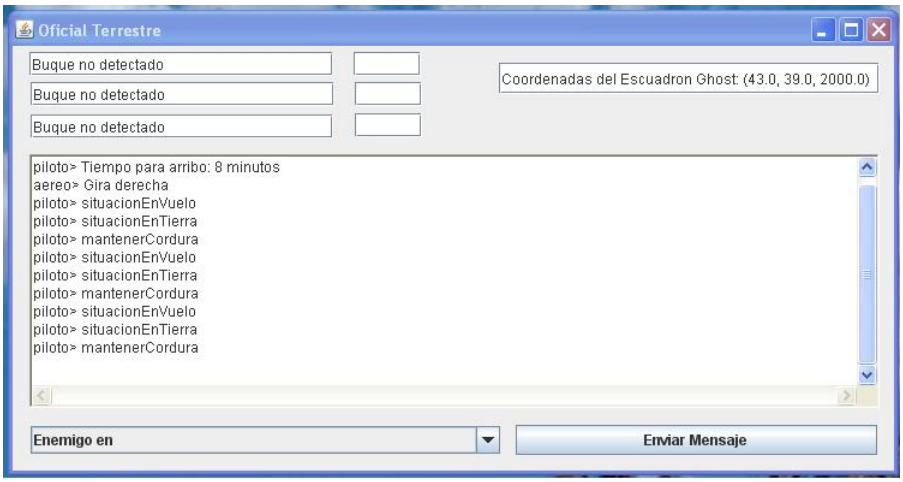

Figura 5.2.4: Interfaz Mando Terrestre *Ghost Squadron*

El mando terrestre recibe información de las coordenadas de los barcos enemigos, siempre que éstos se encuentren dentro de su campo de visión, en el flanco superior izquierdo de su pantalla. A su vez, recibe permanentemente información acerca de la posición del piloto en el flanco superior derecho de la pantalla.

En la parte central mantiene un registro con los mensajes intercambiados entre ambos mandos y el piloto, y finalmente, en la franja inferior posee un selector con los posibles mensajes a emitir y un botón para enviarlos.

5.3. Contextualización de las Creencias

5.3.1. Identificación de las Creencias a Manejar sobre el Contexto Elegido

5.3.1.1. Creencias acerca de Recintos

En el contexto para el entrenamiento un agente podrá mantener información acerca de dos recintos particulares: el primero se corresponde con el área geográfica a través de la cual el escuadrón deberá sortear los vehículos enemigos, la zona enemiga; el segundo recinto, denominado zona segura, representa el lugar hacia el que el escuadrón desea llegar sin ser detectado, o al menos, sin ser derribado.

Las creencias que un agente PANCHO mantiene acerca de dichos recintos, son:

Zona Enemiga:

Características Definitorias:

- Identificador. Nombre único mediante el cual se hará referencia al recinto. En este contexto será *Zona Enemiga*.
- Dimensiones. Definida por dos coordenadas (x, y, z) que indican los extremos opuestos del cubo que representa el espacio virtual por donde se desplazará el escuadrón (el avión).

Zona Segura:

Características Definitorias:

- Identificador. Nombre único mediante el cual se hará referencia al recinto. En este contexto será *Zona Segura*.
- Posición. Definida por dos coordenada (x, y, z) que indican los extremos opuestos del cubo que representa el espacio virtual en donde se encuentra la zona segura.

Estados Transitorios:

- Distancia. Espacio que separa al avión donde se encuentra el agente que mantiene esta creencia, de la zona segura.

5.3.1.2. Creencias acerca de Objetos

En cuanto a los objetos que pueden encontrarse en el entorno: aviones y buques enemigos, así como el avión del escuadrón; las creencias que serán mantenidas por el AVI son:

Avión Enemigo:

Características Definitorias:

- Identificador. Nombre único mediante el cual se distinguirá cada uno de los aviones enemigos (p.e. AVE 01, AVE 02, etcétera).
- Tipo. Clase de objeto del cual se mantiene un conjunto de creencias; en este caso es *vehículo aéreo*.

Estados Transitorios:

- Posición. Conjunto de valores (x, y, z) que señalan el lugar donde se encuentra el objeto.
- Distancia. Espacio que separa al avión donde se encuentra el agente que mantiene esta creencia, del avión enemigo.
- Visible. Indicador de si el avión enemigo se encuentra dentro del campo de visión del avión donde se encuentra el agente que mantiene la creencia.
- Amenaza. Indicador de si el avión enemigo ha emprendido una ofensiva sobre el avión en donde se encuentra el agente.
- Derribado. Creencia acerca de si el avión enemigo ha sido derribado.

Buque Enemigo:

Características Definitorias:

- Identificador. Nombre único mediante el cual se distinguirá cada uno de los buques (p.e. Buque 01, Buque 02, etcétera).
- Tipo. Clase de objeto del cual se mantiene un conjunto de creencias; en este caso es *vehículo marítimo*.

Estados Transitorios:

- Posición. Conjunto de valores (x, y) que señalan el lugar donde se encuentra el objeto (este objeto se mantiene en el plano z=0).
- Distancia. Espacio que separa al avión donde se encuentra el agente que mantiene esta creencia.
- Visible. Indicador de si el buque se encuentra dentro del campo de visión del avión donde se encuentra el agente que mantiene la creencia.
- Amenaza. Indicador de si el buque ha emprendido una ofensiva sobre el avión en donde se encuentra el agente.
- Hundido. Creencia acerca de si el buque ha sido destruido.

Avión del Escuadrón:

Características Definitorias:

- Identificador. Nombre único que distingue el avión del escuadrón del resto de objetos, en este caso de le denomina *Ghost*.
- Tipo. Clase de objeto del cual se mantiene un conjunto de creencias, en este caso es *vehículo aéreo*.

Estados Transitorios:

- Posición. Conjunto de valores (x, y, z) que señalan el lugar donde se encuentra el objeto.
- Perceptible. Indicador de si el avión en el que el agente se desplaza, se encuentra dentro del campo de visión de algún vehículo enemigo.
- En_vuelo. Creencia acerca de si el avión en el cual se encuentra el agente que mantiene la creencia, se mantiene en vuelo (no ha sido derribado).
- Velocidad. Valor numérico que indica la rapidez del vehículo.

5.3.1.3. Creencias acerca de Individuos

Las creencias de tipo características definitorias que un agente PANCHO va a manejar sobre sí mismo (o sobre otros agentes PANCHO del equipo) en nuestra Concreción Contextual, son:

- *Rol de Equipo.* Rol que desempeña principalmente un agente en el entorno, de acuerdo con la clasificación de los nueve tipos de roles propuestos en [Belbin, 1981, 1993]. En esta Concreción Contextual se considerarán sólo dos posibles roles: *Coordinador* e *Implementador,* escogidos de entre los nueve propuestos.
- *Identificador.* Clave o nombre único mediante el cual se diferenciará del resto de individuos que habitan en el entorno.
- *Tipo.* Identificador del personaje que desempeña el agente en el contexto, puede ser: *Piloto, Oficial_Aéreo u Oficial Terrestre.*

Los rasgos de personalidad se establecen a partir del modelo de los cinco factores [Costa and McCrae, Digman and Inouye, 1986]. Dicho modelo considera cinco dimensiones de la personalidad, cada una caracterizada con un conjunto de adjetivos que describen la personalidad que un individuo posee en torno a cierto rasgo. Los factores del modelo son:

- *Extraversión.* Es un rasgo que identifica el compromiso social de un individuo. En un extremo de la escala describe a una persona sociable, conservadora y amigable, y en el otro extremo, una persona introvertida se caracteriza por se tímida, callada y reservada.

- *Amabilidad.* Refleja el grado de capacidad de una persona para mantenerse en armonía con los demás. En un extremo, se encuentran las personas que son amigables y apreciadas por su habilidad para desarrollar y mantener relaciones interpersonales, aunque en ocasiones pueden llegar a ser consideradas ingenuas. En el otro extremo, una persona poco afable puede parecer muy independiente, inflexible y en cierta forma, antipática.

- *Responsabilidad.* Tiene relación con el grado de compromiso y fiabilidad. Una persona escrupulosa con su deber puede ser considerada fiable, organizada y con un fuerte sentido del logro. En el otro extremo, una persona con un bajo nivel de responsabilidad podría tener dificultades para ser perseverante de forma que le permita concluir tareas desafiantes.

- *Estabilidad Emocional.* Es un factor caracterizado por el nivel de ansiedad o de disposición a la preocupación de un individuo. Considerándolo en su ámbito positivo, describe individuos que tienden a ser calmados, emocionalmente estables, y seguros de sí mismos. En el otro extremo de la escala se presenta una tendencia a experimentar emociones negativas como la ansiedad, el enfado o la depresión.

- *Apertura.* Es el factor más difícil de clarificar; los expertos le han asignado varios nombres: cultura, intelecto, apertura intelectual, entre otros. Describe una dimensión que distingue a la gente imaginativa o creativa, de la gente convencional. En un extremo, la gente abierta disfruta de afrontar riesgos, es de mente abierta, imaginativa y con frecuencia inteligente. En el otro extremo, la gente cerrada prefiere la rutina, la tradición, y suele ser reacia al cambio.

Los valores de los rasgos de personalidad están especificados en función del rol de equipo. Los valores que han sido seleccionados para los roles Coordinador e Implementador en la presente concreción, se ilustran en la tabla 5.1.

Rasgos de Personalidad	Rol de Equipo Coordinador	Rol de Equipo Implementador
Extraversión	Bastante	Medianamente
Amabilidad	Bastante	Medianamente
Responsabilidad	Medianamente	Absolutamente
Estabilidad Emocional	Bastante	Medianamente
Apertura	Bastante	Nada

Tabla 5.1: Concreción de los rasgos de personalidad para PANCHO

Por su parte, los estados transitorios que se han considerado para el presente dominio, están vinculados con las emociones que el agente puede presentar durante la realización de la tarea en equipo. Las emociones seleccionadas para el contexto de la aplicación son:

- *Miedo:* Ante la amenaza o riesgo que puede generar una situación no deseada (p.e. la cercanía de un vehículo enemigo).
- *Alegría:* Provocado por algún evento que genere gozo o júbilo (p.e. estar cerca de la zona segura).
- *Sorpresa:* Ante un acontecimiento imprevisto o repentino (p.e. detección de un vehículo enemigo en donde no se esperaba).

Los valores que las emociones pueden tomar en un momento dado van desde *Nada* hasta *Absolutamente*.

Los estados físicos y las actitudes, de acuerdo con el objetivo del contexto de esta aplicación, no requerirán ser considerados. Por su parte, la información vinculada con la posición del propio agente, será considerada como la mantenida por el avión del escuadrón; dicha información le será proporcionada por el EVI.

5.3.1.4. Creencias acerca de la Situación Actual

El cuarto conjunto de creencias para la presente concreción, la situación actual, tiene relación con el estado de seguridad que guarda en cada instante el escuadrón. Los cinco posibles estados en los que el agente PANCHO puede estar en cualquier momento del entrenamiento son:

1. Seguro. El agente se encuentra en zona segura.
2. Alerta. El agente se encuentra en zona enemiga, pero no existe enemigo alguno a la vista.
3. Peligro. El agente se encuentra en zona enemiga, y se ha identificado a un enemigo.
4. Conflicto. El agente se encuentra en zona enemiga, y un vehículo enemigo ha identificado al escuadrón.
5. Derribado. El avión del escuadrón ha sido derribado como consecuencia de un ataque enemigo.

La figura 5.3.1 ilustra los posibles estados y transiciones de la situación que mantiene un agente de tipo PANCHO.

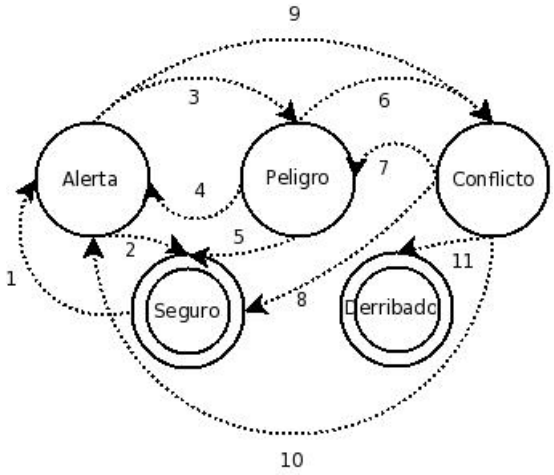

Figura 5.3.1: Estados posibles de la Situación Actual en el EVI

Las transiciones entre los posibles estados durante el proceso de entrenamiento, son los siguientes:

1. Ingreso a la zona enemiga.
2. Ingreso a la zona segura, tras haber transitado por la zona de peligro sin haberse encontrado con vehículos enemigos.
3. Se ha detectado (visible) un enemigo.
4. Tras haber identificado a un enemigo, se ha logrado alejar de éste sin haber si detectado.
5. Tras haber identificado a un enemigo, se ha llegado a la zona segura sin haber sido detectado por el enemigo.
6. Tras haber identificado a un enemigo, éste ha logrado detectar el avión del escuadrón (percibido).
7. El avión del escuadrón ha logrado alejarse del enemigo después de que éste le detectara (percibido). No obstante, el enemigo aún continúa en el campo de visión del escuadrón.
8. Tras haber sido detectado por el enemigo, se ha llegado a la zona segura sin haber sido derribado.

9. Sin haber detectado (visible) al enemigo previamente (por un descuido del oficial responsable) éste ha identificado (percibido) el avión del escuadrón.

10. El avión del escuadrón ha logrado alejarse o deshacerse (derribado o hundido) del enemigo, después de que éste le hubiera detectado (percibido). La acción ha sido tan rápida que el escuadrón ha perdido al enemigo de su campo de visión.

11. El avión del escuadrón no ha logrado alejarse o deshacerse del enemigo; como resultado de ello, éste le ha derribado.

Además del estado interno del agente es necesario recordar los sucesos que tienen relación con la última acción de naturaleza social que ha sido ejecutada por algún compañero. Esta información, que forma parte de la situación actual del agente, será manejada por el mecanismo de estado de la comunicación.

Así, la situación actual de un agente, se encuentra definida por:

Situación actual:

- Estado: el estado en el que se encuentre el agente en un momento determinado puede ser: Seguro, Alerta, Peligro, Conflicto o Derribado.
- Estado de la Comunicación: el último mensaje recibido por un agente, puede ser: enemigo_en, ataque_enemigo_en, permiso_para_ofensiva, despejado, tiempo_para_destino, ofensiva_a_criterio, ascender, descender, gira_izquierda, gira_derecha, de_acuerdo, situacion_controlada, en_desacuerdo

5.3.2. Especificación de las Relaciones entre Creencias del Modelo Personal

En esta subsección, especificamos las relaciones entre las creencias específicas que han sido consideradas en este proyecto, como parte de las creencias interrelacionadas del modelo personal para un agente de tipo PANCHO: *Rasgos de Personalidad y Emociones.*

Los rasgos de personalidad pueden promover un aumento o disminución de la intensidad en los estados emocionales de un individuo; dichas relaciones de influencia pueden ser especificadas de la siguiente manera:

5.3.2.1. Rasgos de Personalidad sobre las Emociones

Extraversión ACENTÚA <Bastante> *Alegría*

Extraversión ACENTÚA <Medianamente> *Sorpresa*

Responsabilidad I-BIPOLAR(+) <Bastante> *Miedo*

Estabilidad Emocional I-BIPOLAR(-) <Medianamente> *Miedo*

Apertura ATENÚA <Medianamente> *Sorpresa*

I-BIPOLAR(+). El Influyente puede provocar una variación en el valor del Influido en una magnitud igual al producto del valor del influyente por el grado de influencia. Dicha variación depende del valor del Influyente.

<A> I-BIPOLAR(+) grado <B> :

Si <A> ε {Bastante,Absolutamente} => <B>' = <B> + <A> × grado

Si <A> ε {Medianamente} => <B>' = <B>

Si <A> ε {Nada, Ligeramente} => <B>' = <B> - (1- <A>) × grado

I-BIPOLAR(-). El Influyente puede provocar una variación en el valor del Influido en una magnitud igual al producto del valor del influyente por el grado de influencia. Dicha variación depende del valor del influyente y es contraria a la producida por la Influencia Bipolar positiva.

<A> I-BIPOLAR(+) grado <B> :

Si <A> ε {Bastante,Absolutamente} => <B>' = <B> - <A> × grado

Si <A> ε {Medianamente} => <B>' = <B>

Si <A> ε {Nada, Ligeramente} => <B>' = <B> + (1- <A>) × grado

5.3.3. Determinación del Valor de Reposo y Parámetros de Degradado

Los estados transitorios requieren ser modelados tomando en consideración la permanencia del agente en el EVI bajo la influencia del paso del tiempo.

En PANCHO se ha definido un valor *inicial,* un valor de *aproximación,* y una *función de evolución*, para gestionar el mantenimiento de cada uno de dichos factores. Dado que los estados físicos no requieren ser mantenidos por el agente, serán las emociones las únicas a ser modeladas mediante este mecanismo. Los valores que hemos seleccionado para modelar las emociones con este mecanismo, se presentan en la tabla 5.2.

Emoción	Valor Inicial	Función de Evolución
Miedo	Ligeramente	*Emoción = Emoción * (1.0002)*
Alegría	Ligeramente	*Emoción = Emoción * (0.009)*
Sorpresa	Ligeramente	*Emoción = Emoción * (0.009)*

Tabla 5.2: Valor inicial y función de evolución en PANCHO

El valor de aproximación al que tenderán cada una de las emociones consideradas, dependerá del rol de equipo desempeñado por PANCHO, y a su vez, de los valores seleccionados para sus rasgos de personalidad. Por ello no son asignados a priori, sino que serán calculados al momento de especificar el rol de equipo.

5.4. Intereses del Agente

Los intereses de un agente de tipo PANCHO, de acuerdo con la arquitectura, afectan a un subconjunto de creencias vinculadas con los estados transitorios que el agente mantiene; en el caso de la presente concreción, estos se corresponden con las emociones que PANCHO mantiene en cada momento. En las siguientes subsecciones se establecen los umbrales para dichas creencias, así como las relaciones de influencia que los rasgos de personalidad mantienen sobre éstos.

5.4.1. Determinación de los Umbrales de los Intereses

Los valores seleccionados para los umbrales inferior y superior, para las emociones en un agente de tipo PANCHO, han sido determinados sin considerar el tipo de rol que el agente desempeñará en el equipo. No obstante, para la generación de sus comportamientos, el valor en las emociones mantenido por el agente, se encuentra influenciado por los valores que hayan sido elegidos para los rasgos de personalidad.

Para la presente aplicación, los valores seleccionados para los intereses en torno a los umbrales de las emociones en cada uno de los dos posibles roles de equipo que PANCHO puede asumir, se presentan en la tabla 5.3.

5.4.2. Relaciones entre Rasgos de Personalidad e Intereses

Entre cada rasgo de personalidad y cada uno de los intereses mantenidos por el agente, es posible la existencia de dos relaciones como máximo, una para cada uno de los umbrales. No obstante, la existencia de la influencia de un rasgo de personalidad sobre uno de los umbrales, no implica la existencia de influencia en el otro.

5.4.2.1. Rasgos de Personalidad sobre Umbrales Superiores (USup) de los Intereses

Responsabilidad ACENTÚA <Ligeramente> *Miedo*

Estabilidad Emocional ACENTÚA <Medianamente> *Miedo*

Apertura ACENTÚA <Medianamente> *Sorpresa*

En el caso del miedo y de la sorpresa, el interés del agente radica en permitir elevar el nivel máximo (umbral superior) aceptable, que un agente puede controlar durante la misión.

5.4.2.2. Rasgos de Personalidad sobre Umbrales Inferiores de los Intereses

Extraversión ATENÚA <Ligeramente> *Alegría*

En este segundo caso, el interés del agente radica en aumentar el rango en el nivel mínimo (umbral inferior) de alegría que un agente podría mantener, para poder tener un desempeño aceptable en la misión.

5.5. Acciones Ejecutables por el Agente

Las acciones que un agente de tipo PANCHO puede ejecutar, en función de su tratamiento en la arquitectura, pueden ser *concreta*s o *abstractas*. En las acciones abstractas será necesario especificar el tratamiento interno (razonamiento) que deberá dárselas para concretarlas adecuadamente.

Emoción	U.Inf (Coordinador)	U.Sup (Coordinador)	U.Inf (Implementador)	U.Sup (Implementador)
Miedo	Ligeramente	Absolutamente	Nada	Bastante
Alegría	Ligeramente	Absolutamente	Nada	Bastante
Sorpresa	Ligeramente	Absolutamente	Nada	Medianamente

Tabla 5.3: Umbrales inferior y superior para los Intereses en las Emociones

Por otro lado, en función de la naturaleza de la acción a realizar, distinguiremos entre acciones de *naturaleza técnica* y acciones de *naturaleza social*. En la subsección 5.5.1 se especifican las acciones abstractas y concretas de naturaleza técnica ejecutables por PANCHO, bajo el esquema definido en COGNITIVA. Por su parte, las acciones de naturaleza social que pueden ser realizadas por cualquier miembro del grupo (humano o agente) en el EVI, se especifican en la subsección 5.5.2.

5.5.1. Acciones de Naturaleza Técnica

Pese a que se definirán todas las acciones técnicas de todos los miembros del equipo, tan sólo se detallarán en profundidad las que realiza el piloto. Es así debido a que en las pruebas realizadas fue el piloto el miembro del equipo controlado por un agente virtual, mientras que los otros dos eran humanos. Así mismo, la actuación del agente responderá a los estilos previamente descritos para los roles de equipo Coordinador e Implementador.

5.5.1.1. Acciones del Piloto

Las acciones vinculadas con la tarea que un agente con el personaje Piloto puede realizar en el entorno son:

Ascender:

Precondiciones:

- = (En_vuelo(Ghost), Sí)
- <= (posición_z(Ghost), 11000m)

Operador:

- ascender (cantidad)

Consecuencias:

- = (posición_z (Ghost), z + cantidad)

Expectativas:

- Acontecimiento: = (posición_z (Ghost), z + cantidad)
    - Expectación: <Absolutamente>
    - Deseo: <Absolutamente>
- Acontecimiento: ≠ (posición_z (Ghost), z + cantidad)
    - Expectación: <Ligeramente>
    - Deseo: <Nada>

Indicador de caducidad:

- 20 segundos.

Descender:

Precondiciones:

- = (En_vuelo(Ghost), Sí)
- >= (posición_z(Ghost), 2000m)

Operador:

- descender (cantidad)

Consecuencias:

- = (posición_z (Ghost), z - cantidad)

Expectativas:

- Acontecimiento: = (posición_z (Ghost), z - cantidad)
    - Expectación: <Absolutamente>
    - Deseo: <Absolutamente>

- Acontecimiento: ≠ (posición_z (Ghost), z - cantidad)
    - Expectación: <Ligeramente>
    - Deseo: <Nada>

Indicador de caducidad:

- 20 segundos.

Las acciones ascender y descender tienen como propósito pilotar el avión del escuadrón para generar un incremento, o decremento, en el valor de su altitud (valor en z). El agente espera que una vez ejecutada la acción, el avión haya modificado su altitud en la cantidad especificada; no obstante, pudiese ocurrir que algún suceso le impidiese tal modificación, o dicho de otra manera, que la ejecución de la acción se hubiese malogrado.

Girar_Derecha:

Precondiciones:

- = (En_vuelo(Ghost), Sí)

Operador:

- girar_derecha(cantidad)

Consecuencias:

- = (posición_x (Ghost), x + 1)
- = (posición_y (Ghost), y + cantidad)

Expectativas:

- Acontecimiento: = (posición_x (Ghost), x + 1) ^ = (posición_y (Ghost), y + cantidad)
    - Expectación: <Absolutamente>

- Deseo: <Absolutamente>

- Acontecimiento: ≠ (posición_x (Ghost), x + 1) ^ = (posición_y (Ghost), y + cantidad)
  - Expectación: <Ligeramente>
  - Deseo: <Nada>

Indicador de caducidad:

- 20 segundos.

Girar_Izquierda:

Precondiciones:

- = (En_vuelo(Ghost), Sí)

Operador:

- girar_izquierda(cantidad)

Consecuencias:

- = (posición_x (Ghost), x + 1)
- = (posición_y (Ghost), y - cantidad)

Expectativas:

- Acontecimiento: = (posición_x (Ghost), x + 1) ^ = (posición_y (Ghost), y - cantidad)
  - Expectación: <Absolutamente>
  - Deseo: <Absolutamente>

- Acontecimiento: ≠ (posición_x (Ghost), x + 1) ^ = (posición_y (Ghost), y - cantidad)
  - Expectación: <Ligeramente>
  - Deseo: <Nada>

Indicador de caducidad:

- 20 segundos.

El propósito de las acciones girar_derecha y girar_izquierda, es el de pilotar el avión del escuadrón para modificar su dirección durante el desplazamiento que realiza en vuelo; en ambos casos, el desplazamiento se realiza en una cierta cantidad pasada como parámetro, en el eje de las y (a la izquierda, o a la derecha de la posición original). Sin embargo, dado que el avión se mantiene en vuelo, su desplazamiento será también en una unidad en el eje x (hacia adelante).

Eludir_enemigo:

Precondiciones:

- = (RolEquipo(Pancho), Coordinador)
- = (Estado(Pancho), Peligro)

Operador:

- eludir_enemigo()

Consecuencias:

- = (Estado(Pancho), Alerta)

Expectativas:

- Acontecimiento: = (Visible(Ghost), No)

- Expectación: <Absolutamente>
- Deseo: <Absolutamente>

- Acontecimiento: = (Visible(Ghost), Si)
  - Expectación: <Ligeramente>
  - Deseo: <Nada>

Indicador de caducidad:

- 30 segundos.

En este caso, la acción *Eludir_Enemigo*, podrá ser ejecutada únicamente en el caso en que el agente PANCHO desempeñe el rol de equipo Coordinador. Esto responde a la intención de diferencial el comportamiento de un líder, el cual ante una situación que podría poner en riesgo al equipo, es capaz de anticiparse a un suceso y tomar una decisión que involucra a todo el escuadrón (aunque no sea la más adecuada); por su parte el Implementador suele esperar indicaciones de actuación ante una situación que no se encuentra claramente definida.

Eludir_enemigo es una acción abstracta, y se podrá ejecutar siempre que el estado de la situación actual sea *Peligro*; la acción tiene como objetivo alejarse del área de visibilidad del enemigo, y para ello deberá resolver en tiempo de ejecución, la(s) acciones concretas a ejecutar.

La acción abstracta eludir_enemigo se descompone en:

1. girar_sentido(),
2. eludir_enemigo().

Donde girar_sentido sería sustituido por la acción más adecuada entre las acciones girar_derecha o girar_izquierda, con el objetivo de alejar al escuadrón de la posición en la que ha sido detectado el enemigo, es decir:

- Si posición_x (Enemigo) > = posición_x (Ghost) entonces girar_izquierda()
- Si posición_x (Enemigo) < posición_x (Ghost) entonces girar_derecha()

Huir:

Precondiciones:

- = (Estado(Pancho), Conflicto)

Operador:

- huir()

Consecuencias:

- = (Estado(Pancho), Peligro)

Expectativas:

- Acontecimiento: = (Perceptible(Enemigo), No)
  - Expectación: <Absolutamente>
  - Deseo: <Absolutamente>
- Acontecimiento: = (Perceptible(Enemigo), Si)
  - Expectación: <Ligeramente>
  - Deseo: <Nada>

Indicador de caducidad:

- 30 segundos.

La acción *Huir* es también una acción abstracta, y se podrá ejecutar siempre que el estado de la situación actual sea *Conflicto*; la acción tiene como objetivo alejarse del área de percepción del enemigo, y para ello deberá, resolver en tiempo de ejecución, las acciones concretas a ejecutar.

Debido a la ausencia de incertidumbre respecto a la posible situación de riesgo del escuadrón (el riesgo es real ya que se encuentra en una situación de conflicto), esta acción podrá

ser ejecutada por cualquiera de los dos roles de equipo. En este caso, la diferencia en el estilo de ejecución de un Coordinador frente a la de un Implementador, radica en que el segundo tendrá una combinación de acciones concretas más elaborada (más eficiente en cuanto al propósito de la acción).

Si PANCHO desempeña el rol de Coordinador, la acción huir se descompondría con un mecanismo similar al de eludir_enemigo; sin embargo, si la accion abstracta pretende ser ejecutada por el rol Implementador, la acción huir presentaría la siguiente descomposición:

1. girar_sentido(),
2. modificar_altitud(),
3. huir().

Donde girar_sentido sería sustituido por la acción más adecuada entre girar_derecha o girar_izquierda, de forma que le aleje de la posición en la que ha sido detectado el enemigo (el mismo procedimiento que en eludir enemigo). Por su parte, modificar_altitud sería sustituido por la acción más adecuada entre las acciones ascender o descender, en función del personaje que le haya avisado del ataque enemigo, es decir:

- Si Oficial_terrestre = emisor (mensaje ('Ataque Enemigo en X, Y, Z')) entonces ascender()
- Si Oficial_aéreo = emisor (mensaje ('Ataque Enemigo en X, Y, Z')) entonces descender()

5.5.1.2. Acciones de los otros Personajes

Las acciones vinculadas con la tarea, que tendrían que ser implementadas en el caso de que un agente virtual desempeñara el personaje de Oficial Aéreo, o el de Oficial Terrestre, son:

- *Vigilar()*. Funcionalidad mediante la cual el agente se mantiene pendiente del horizonte de visión (aéreo o terrestre) en todo momento. La expectativa deseable es no encontrarse con enemigos durante el viaje, no obstante, puede presentarse el acontecimiento de detectar a un enemigo, dicha situación modificaría la situación actual de Alerta a Peligro.

- *Derribar()*. Procedimiento mediante el cual el Oficial correspondiente inicia un ataque contra un enemigo, con la expectativa de derribarlo o hundirlo.

5.5.2. Acciones de Naturaleza Social

Las acciones de naturaleza social establecidas para el escenario están relacionadas con actos de comunicación tendientes al logro del objetivo de la misión, y acordes con la dinámica establecida. La tabla lista las acciones establecidas para el escenario, así como los personajes que podrán hacer uso de éstas.

Acción (Verbal)	Piloto	Oficial Aéreo	Oficial Terrestre
Permiso para ofensiva		X	X
Enemigo en (X,Y,Z)		X	X
Situación en vuelo?	X		
Situación en tierra?	X		
Tiempo para destino?		X	X
Despejado		X	X
Tiempo para arribo (Minutos)	X		
Ofensiva a criterio	X	X	X
Ascender		X	X
Descender		X	X
Gira derecha		X	X
Gira izquierda		X	X
De acuerdo		X	X
Mantener la cordura	X		
Situación controlada		X	X
Trayecto recorrido (Porcentaje)	X		
En desacuerdo		X	X
Ataque enemigo en (X, Y, Z)		X	X

Tabla 5.4: Acciones de naturaleza social en *Ghost Squadron*

Debido a que hemos seleccionado el personaje Piloto para ser ejecutado por el agente virtual PANCHO, en la presente concreción especificaremos sólo las acciones de naturaleza social que pueden ser ejecutadas por dicho personaje; no obstante, para evitar interpretaciones erróneas en la dinámica de comunicación que ha sido concebida para el EVI, ofreceremos también una breve descripción del propósito del resto de acciones.

5.5.2.1. Acciones del Piloto

Situación en vuelo?:

Precondiciones:

- = (Estado(Pancho), Peligro)

Operador:

- mensaje("Situación en vuelo?")

Consecuencias:

- Ninguna

Expectativas:

- Acontecimiento: = (Estado_Comunicación (Oficial_Aéreo), despejado)
  - Expectación: <Bastante>
  - Deseo: <Absolutamente>
- Acontecimiento: = (Estado_Comunicación (Oficial_Aéreo), enemigo_en)
  - Expectación: <Ligeramente>
  - Deseo: <Nada>

Indicador de caducidad:

- 5 segundos.

Situación en tierra?:

Precondiciones:

- = (Estado(Pancho), Peligro)

Operador:

- mensaje("Situación en tierra?")

Consecuencias:

- Ninguna

Expectativas:

- Acontecimiento: = (Estado_Comunicación (Oficial_Terrestre), despejado)
  - Expectación: <Bastante>
  - Deseo: <Absolutamente>
- Acontecimiento: = (Estado_Comunicación (Oficial_Terrestre), enemigo_en)
  - Expectación: <Ligeramente>
  - Deseo: <Nada>

Indicador de caducidad:

- 5 segundos.

Tras haber ocurrido un acontecimiento que ha dejado en un estado de Peligro al situación del escuadrón, y habiendo modificado previamente la ruta del vuelo, el piloto solicita información respecto de la proximidad del enemigo. Aunque esta última acción (social, por tanto, un mensaje) no genera una consecuencia en el EVI, la expectativa deseable es que el escuadrón ya no se encuentre al alcance del enemigo, no obstante se tiene cierta expectativa de que no sea así.

Tiempo para arribo:

Precondiciones:

- = (Estado(Pancho), Alerta)

- = (Estado_Comunicación(Oficial), tiempo_para_destino)

Operador:

- mensaje("Tiempo para arribo", Minutos)

Consecuencias:

- Ninguna

Expectativas:

- Acontecimiento: = (Estado_Comunicación (Oficial), de_acuerdo)
  - Expectación: <Bastante>
  - Deseo: <Bastante>

Indicador de caducidad:

- 5 segundos.

Ofensiva a criterio:

Precondiciones:

- = (Estado(Pancho), Conflicto)
- = (Estado_Comunicación(Oficial), permiso_para_ofensiva)

Operador:

- mensaje("Ofensiva a criterio")

Consecuencias:

- Ninguna

Expectativas:

- Acontecimiento: = (Estado_Comunicación (Oficial), de_acuerdo)
  - Expectación: <Bastante>
  - Deseo: <Bastante>
- Acontecimiento: = (Estado_Comunicación (Oficial), en_desacuerdo)
  - Expectación: <Ligeramente>
  - Deseo: <Nada>

Indicador de caducidad:

- 5 segundos.

Se envía "Ofensiva a criterio" como respuesta a una solicitud para proceder con una acción, que en este caso se corresponde con el ataque a un enemigo presente dentro del campo de visión del escuadrón. Se tiene un grado de expectación elevado, así como deseo, por la aceptación por parte del otro miembro del equipo, pero sobre todo, una absoluta expectación de ejecutar el ataque. Existe también una ligera preocupación por la desaprobación de la acción por parte de algún compañero.

Mantener la cordura:

Precondiciones:

- = (RolEquipo(Pancho), Coordinador)
- = (Estado(Pancho), Peligro)

Operador:

- mensaje("Mantener la cordura")

Consecuencias:

- Ninguna

Expectativas:

- Acontecimiento: = (Estado_Comunicación (Oficial), de_acuerdo)
  - Expectación: <Bastante>
  - Deseo: <Bastante>

Indicador de caducidad:

- 10 segundos.

Se envía para intentar liberar la tensión en el equipo, pero únicamente cuando el agente PANCHO desempeñe el rol de Coordinador.

Trayecto recorrido:

Precondiciones:

- = (RolEquipo(Pancho), Coordinador)
- <= (Distancia(Zona Segura), 5 %)

Operador:

- mensaje("Trayecto recorrido", P %)

Consecuencias:

- Ninguna

Expectativas:

- Acontecimiento: = (Estado_Comunicación (Oficial), de_acuerdo)
  - Expectación: <Bastante>

- Deseo: <Bastante>

Indicador de caducidad:

- 5 segundos.

Enviado para liberar la tensión en el equipo, cuando éste haya recorrido al menos el 95 % del trayecto. Este mensaje solamente será ejecutado por PANCHO, cuando desempeñe el rol de equipo Coordinador.

5.5.2.2. Acciones de otros Personajes

Las acciones vinculadas con los mensajes, que tendrían que ser implementadas en el caso de que un agente virtual desempeñara el personaje de Oficial Aéreo, o el de Oficial Terrestre, son:

- *Permiso para ofensiva.* Ante un estado de conflicto, y estando a una distancia en la que el avión puede ser derribado por el enemigo, el emisor (oficial aéreo o terrestre) del mensaje solicita la opinión de sus compañeros para emprender una acción ofensiva. El emisor del mensaje mantiene la expectativa de que se le otorgue la libertad de atacar al enemigo, sin embargo, no descarta una posible negativa.
- *Enemigo en X, Y, Z.* Indica una situación que propicia un cambio en el estado de la situación actual para el agente (dejándole en peligro) y el emisor queda en espera de posibles sugerencias o acciones por realizar para modificar dicho estado. Una alternativa para su utilización es que, ante un estado aún de peligro, el emisor ofrezca orientación respecto de la posición actual del enemigo.
- *Tiempo para destino?* Estando en estado de Alerta, y luego de haber transcurrido cierto tiempo y acontecimientos en el EVI, la siguiente acción, tiene como propósito reducir la incertidumbre sobre la finalización de la misión.
- *Despejado*. Como resultado de un mensaje solicitando la situación del escuadrón, el siguiente mensaje genera un cambio en el estado de la situacion actual, hacia un estado de Alerta. Se tiene el deseo de que el mando terrestre confirme dicha situación, aunque no se descarta la posibilidad de un enemigo cercano.

- *Ascender, Descender, Girar_derecha y Girar_izquierda.* Mensajes enviados como sugerencia al Piloto, tras haberse detectado un enemigo por parte del oficial aéreo o terrestre. En los cuatro casos, se tiene la expectativa de iniciar una acción que responda a dicha indicación.

- *De acuerdo*. Mensaje con el que se muestra la aprobación de la intención expresada en un mensaje previo, o de una acción sugerida.

- *Situación controlada.* Mensaje indicativo de que el riesgo latente, por estar en una situación de conflicto o de peligro, ha sido solucionado. La intensión del emisor es liberar la tensión mantenida por el grupo.

- *En desacuerdo.* Mensaje con el que se muestra la desaprobación de la intención expresada en un mensaje previo, o de una acción sugerida.

- *Ataque enemigo en X, Y, Z.* Mensaje con el que se muestra cierta tensión ante la existencia de un riesgo de supervivencia para el equipo. Este mensaje indica que el estado del escuadrón es ya de Conflicto.

5.6. Identificación de las Expectativas

5.6.1. Expectativas procedentes de las acciones de naturaleza técnica

Tal y como fue establecido en COGNITIVA, una expectativa se compone de un *acontecimiento,* un grado de *expectación*, un grado de *deseo* y un *periodo de validez*.

En la presente concreción, utilizaremos las acciones que hemos definido para el AVI, para identificar todos los acontecimientos esperados, asignándoles como indicador de caducidad, el periodo de validez definido para la acción. Las expectativas vinculadas con las acciones de naturaleza técnica son:

Ascenso:

(Procedente de la acción ascender):

- Acontecimiento:
  - = (posición_z (Ghost), z + cantidad)

- Expectación:
  - <Absolutamente>
- Deseo:
  - <Absolutamente>
- Periodo de validez:
  - 20 segundos.

Ascenso infructuoso:

(Procedente de la acción ascender):

- Acontecimiento:
  - $\neq$ (posición_z (Ghost), z + cantidad)
- Expectación:
  - <Ligeramente>
- Deseo:
  - <Nada>
- Periodo de validez:
  - 20 segundos.

Descenso:

(Procedente de la acción descender):

- Acontecimiento:

- = (posición_z (Ghost), z - cantidad)

- Expectación:
  - <Absolutamente>
- Deseo:
  - <Absolutamente>
- Periodo de validez:
  - 20 segundos.

Descenso infructuoso:

(Procedente de la acción descender):

- Acontecimiento:
  - $\neq$ (posición_z (Ghost), z - cantidad)
- Expectación:
  - <Ligeramente>
- Deseo:
  - <Nada>
- Periodo de validez:
  - 20 segundos.

Giro_a_derecha:

(Procedente de la acción girar_derecha):

- Acontecimiento:
  - = (posición_x (Ghost), x + 1)
  - = (posición_y (Ghost), y + cantidad)
- Expectación:
  - <Absolutamente>
- Deseo:
  - <Absolutamente>
- Periodo de validez:
  - 20 segundos.

Giro_a_derecha infructuoso:

(Procedente de la acción girar_derecha):

- Acontecimiento:
  - $\neq$ (posición_x (Ghost), x + 1)
  - $\neq$ (posición_y (Ghost), y + cantidad)
- Expectación:
  - <Ligeramente>
- Deseo:
  - <Nada>
- Periodo de validez:
  - 20 segundos.

Giro_a_izquierda:

(Procedente de la acción girar_izquierda):

- Acontecimiento:
  - = (posición_x (Ghost), x + 1)
  - = (posición_y (Ghost), y - cantidad)
- Expectación:
  - <Absolutamente>
- Deseo:
  - <Absolutamente>
- Periodo de validez:
  - 20 segundos.

Giro_a_izquierda infructuoso:

(Procedente de la acción girar_izquierda):

- Acontecimiento:
  - $\neq$ (posición_x (Ghost), x + 1)
  - $\neq$ (posición_y (Ghost), y - cantidad)
- Expectación:
  - <Ligeramente>
- Deseo:
  - <Nada>
- Periodo de validez:

- 20 segundos.

Escabullirse:

(Procedente de la acción eludir enemigo):

- Acontecimiento:
  - = (Visible (Ghost), No)
- Expectación:
  - <Absolutamente>
- Deseo:
  - <Absolutamente>
- Periodo de validez:
  - 30 segundos.

Escabullirse infructuoso:

(Procedente de la acción eludir enemigo):

- Acontecimiento:
  - = (Visible (Ghost), Si)
- Expectación:
  - <Ligeramente>
- Deseo:
  - <Nada>

- Periodo de validez:
    - 30 segundos.

Escapar:

(Procedente de la acción huir):

- Acontecimiento:
    - = (Perceptible (Enemigo), No)
- Expectación:
    - <Absolutamente>
- Deseo:
    - <Absolutamente>
- Periodo de validez:
    - 30 segundos.

Escapar infructuoso:

(Procedente de la acción huir):

- Acontecimiento:
    - = (Perceptible (Enemigo), Si)
- Expectación:
    - <Ligeramente>
- Deseo:

- <Nada>

- Periodo de validez:
  - 30 segundos.

5.6.2. Expectativas procedentes de las acciones de naturaleza social

Por otro lado, las expectativas vinculadas con las acciones de naturaleza social son:

Visibilidad despejada:

(Procedente de la acción "situación en vuelo?" o "situación en tierra?"):

- Acontecimiento:
  - = (Estado_Comunicación (Oficial), despejado)
- Expectación:
  - <Bastante>
- Deseo:
  - <Absolutamente>
- Periodo de validez:
  - 5 segundos.

Visibilidad no despejada:

(Procedente de la acción "situación en vuelo?" o "situación en tierra?"):

- Acontecimiento:
  - = (Estado_Comunicación (Oficial), enemigo_en)

- Expectación:
  - <Ligeramente>
- Deseo:
  - <Nada>
- Periodo de validez:
  - 5 segundos.

Aceptación:

(Procedente de alguna de las acciones: “Tiempo para arribo…”, “Ofensiva a criterio”, “Mantener la cordura” y “Trayecto recorrido…”):

- Acontecimiento:
  - = (Estado_Comunicación (Oficial), de_acuerdo)
- Expectación:
  - <Bastante>
- Deseo:
  - <Bastante>
- Periodo de validez:
  - 5 segundos.

Rechazo:

(Procedente de la acción “Ofensiva a criterio”):

- Acontecimiento:

- = (Estado_Comunicación (Oficial), en_desacuerdo)

- Expectación:
  - <Ligeramente>
- Deseo:
  - <Nada>
- Periodo de validez:
  - 5 segundos.

5.6.3. Generación de Emociones a partir de las Expectativas

Debido a que hemos planteado dos posibles personalidades diferentes para el AVI: Coordinador e Implementador, identificaremos de manera separada (para cada uno de los dos roles de equipo) la generación de las emociones a partir de las expectativas.

Así, en las tablas 5.5, 5.6, 5.7 se presenta el mecanismo mediante el cual se verán afectadas las emociones de un agente de tipo PANCHO con el rol de coordinador ante posibles situaciones: no confirmada, cumplida y no cumplida, respectivamente.

Como se podrá observar, en las tablas 5.5 y 5.7, la variación en las emociones generada por la no confirmación, o el no cumplimiento de acontecimientos no esperados, no ha sido considerada en la presente concreción; su inclusión supondría que PANCHO debería responder a cambios emocionales por algo no ocurrido y que no se espera que ocurra.

	Deseable	No Deseable
Esperado	Miedo'=Miedo-Ligeramente Alegría'=Alegría+Medianamente	Miedo'=Miedo+Ligeramente Alegría'=Alegría-Medianamente
No Esperado	-	-

Tabla 5.5: Variación de las emociones en un Coordinador a partir de expectativas no confirmadas

En la tabla 5.8 y en la 5.10, la variación de las emociones generadas por la no ocurrencia o no confirmación de sucesos no esperados por PANCHO, tampoco ha sido considerada.

Las tablas 5.8, 5.9 y 5.10 plantean el mecanismo que será utilizado en PANCHO, en su rol de Implementador.

Se ha diferenciado entre los dos roles de equipo proponiendo variaciones en sus emociones diferentes, vinculadas a su vez con los rasgos de personalidad propios del rol. Así, en el Coordinador, su grado de *alegría* ante las tres posibles situaciones, varía con una mayor magnitud (al ser más extrovertido) en comparación con el Implementador. No obs-

	Deseable	No Deseable
Esperado	Miedo'=Miedo-Ligeramente Alegría'=Alegría+Bastante Sorpresa'=Sorpresa+Ligeramente	Miedo'=Miedo+Ligeramente Alegría'=Alegría-Medianamente Sorpresa'=Sorpresa+Ligeramente
No Esperado	Miedo'=Miedo-Ligeramente Alegría'=Alegría+Absolutamente Sorpresa'=Sorpresa+Medianamente	Miedo'=Miedo+Ligeramente Alegría'=Alegría-Bastante Sorpresa'=Sorpresa+Medianamente

Tabla 5.6: Variación de las emociones en un Coordinador a partir de expectativas cumplidas

	Deseable	No Deseable
Esperado	Miedo'=Miedo-Ligeramente Alegría'=Alegría-Bastante	Miedo'=Miedo+Ligeramente Alegría'=Alegría+Bastante
No Esperado	-	-

Tabla 5.7: Variación de las emociones en un Coordinador a partir de expectativas no cumplidas

	Deseable	No Deseable
Esperado	Miedo'=Miedo-Ligeramente Alegría'=Alegría+Ligeramente	Miedo'=Miedo+Medianamente Alegría'=Alegría-Ligeramente
No Esperado		

Tabla 5.8: Variación de las emociones en un Implementador a partir de expectativas no confirmadas

	Deseable	No Deseable
Esperado	Miedo'=Miedo-Medianamente Alegría'=Alegría+Ligeramente Sorpresa'=Sorpresa+Medianamente	Miedo'=Miedo+Bastante Alegría'=Alegría-Ligeramente Sorpresa'=Sorpresa+Medianamente
No Esperado	Miedo'=Miedo-Medianamente Alegría'=Alegría+Medianamente Sorpresa'=Sorpresa+Bastante	Miedo'=Miedo+Bastante Alegría'=Alegría-Medianamente Sorpresa'=Sorpresa+Bastante

Tabla 5.9: Variación de las emociones en un Implantador a partir de expectativas cumplidas

tante, se ha deseado caracterizar como un individuo menos *miedoso* (por ser más estable emocionalmente). Por su parte, la *sorpresa* se ha utilizado como expresión de la estabilidad emocional y la apertura; de forma que un Coordinador es un individuo más estable que controla su comportamiento ante algún acontecimiento ya confirmado que le cause sorpresa.

5.7. Selección de los Eventos y Perceptos

5.7.1. Descripción de los Eventos Percibidos por los Sensores

En referencia a las características del entorno, pero sobre todo, a la dinámica de interacción entre los miembros del equipo, los eventos que podrán ser percibidos por lo sensores (y analizados en el módulo cognitivo), son los siguientes:

- Un miembro del equipo ha solicitado permiso para atacar.
- Un miembro del equipo ha informado sobre la visibilidad de un enemigo.
- Un miembro del equipo ha solicitado información respecto del tiempo que falta para llegar al destino.
- Un miembro del equipo ha informado sobre la nula visibilidad de enemigo alguno.
- Un miembro del equipo ha sugerido realizar un ascenso durante el vuelo.
- Un miembro del equipo ha sugerido realizar un descenso durante el vuelo.
- Un miembro del equipo ha sugerido realizar un giro a la derecha durante el vuelo.
- Un miembro del equipo ha sugerido realizar un giro a la izquierda durante el vuelo.
- Un miembro del equipo ha dado su aprobación sobre el tema en discusión.

	Deseable	No Deseable
Esperado	Miedo'=Miedo-Bastante Alegría'=Alegría-Ligeramente	Miedo'=Miedo+Bastante Alegría'=Alegría+Ligeramente
No Esperado		

Tabla 5.10: Variación de las emociones en un Implementador a partir de expectativas no cumplidas

- Un miembro del equipo ha informado de que la situación de riesgo ha sido controlada.
- Un miembro del equipo ha presentado su desaprobación sobre el tema en discusión.
- Un miembro del equipo ha informado que el avión del escuadrón ha sido percibido y que se encuentra en riesgo de ataque.

5.7.2. Descripción de los Perceptos

Los *eventos* son traducidos a *perceptos* por el intérprete y dirigidos a los diferentes niveles del módulo cognitivo. Los perceptos manejados por el intérprete en la siguiente concreción serán:

- Permiso_para_Ofensiva (identificador) (Un compañero con personaje *identificador,* ha solicitado permiso para atacar).
- Enemigo_en_X_Y_Z (identificador, x, y, z) (Un compañero con personaje *identificador,* ha informado sobre la posición (en x, y, z) de un enemigo).
- Tiempo_para_destino? (identificador) (Un compañero con personaje *identificador,* ha solicitado información respecto del tiempo estimado que falta para llegar a destino).
- Despejado (identificador) (Un compañero con personaje *identificador,* ha informado de la ausencia de enemigos.
- Ascenso (identificador) (Un compañero con personaje *identificador,* ha sugerido realizar la acción Ascender durante el vuelo).
- Descenso (identificador) (Un compañero con personaje *identificador,* ha sugerido realizar la acción Descender durante el vuelo).
- Girar_derecha (identificador) (Un compañero con personaje *identificador,* ha sugerido realizar la acción girar_derecha durante el vuelo).
- Girar_izquierda (identificador) (Un compañero con personaje *identificador,* ha sugerido realizar la acción girar_izquierda durante el vuelo).
- Aprobar (identificador) (Un compañero con personaje *identificador,* ha dado su aprobación sobre el tema en discusión).
- Situación_Controlada (identificador) (Un compañero con personaje *identificador,* ha informado que la situación de riesgo ha sido controlada).

- Desaprobar (identificador) (Un compañero con personaje *identificador,* ha presentado su desaprobación sobre el tema en discusión).
- Ataque_Enemigo_en_X_Y_Z (identificador, x, y, z) (Un compañero con personaje *identificador,* ha informado que el avión del escuadrón ha sido percibido, y se encuentra en riesgo de ataque por el enemigo que se encuentra en la posición x, y, z.

5.7.3. Correspondencia entre Perceptos y Creencias

En esta subsección se identificará el proceso de traducción y distribución de los perceptos hacia el resto de componentes del módulo cognitivo del agente. En concreto, a partir de los perceptos, el intérprete generará unas determinadas creencias y a su vez, podrán servir como disparador de alguno de los procesos de razonamiento de las distintas capas del AVI. La tabla 5.11 resume la correspondencia identificada entre perceptos y creencias.

PERCEPTOS	CREENCIAS
Permiso_para_Ofensiva (identificador)	Perceptible(Ghost)=Si Estado(PANCHO)=Conflicto Estado_Comunicación(Oficial)=permiso_para_ofensiva
Enemigo_en_X_Y_Z (identificador, x, y, z)	Visible(Enemigo)=Si Estado(PANCHO)=Peligro Posición(Enemigo)=(x,y,z) Posición(Ghost)=(x',y',z') Estado_Comunicación(Oficial)=enemigo_en
Tiempo_para_destino? (identificador)	En_vuelo(Ghost)=Si Estado_Comunicación(Oficial)=tiempo_para_destino
Despejado (identificador)	Perceptible(Ghost)=No Visible(Enemigo)=No Estado(PANCHO)=Alerta Estado_Comunicación(Oficial)=despejado
Ascenso (identificador) *	Posición(Ghost)=(x,y,z) Estado_Comunicación(Oficial)=ascender
Aprobar (identificador)	Estado_Comunicación(Oficial)=de_acuerdo
Situación_Controlada (identificador)	Miedo'=Miedo-Ligeramente Si Rol (PANCHO) = Coordinador Entonces Alegría'=Alegría+Bastante Sorpresa'=Sorpresa+Ligeramente Si Rol (PANCHO) = Implementador Entonces Alegría'=Alegría+Ligeramente Sorpresa'=Sorpresa+Medianamente Estado_Comunicación(Oficial)=situacion_controlada
Desaprobar (identificador)	Estado_Comunicación(Oficial)=en_desacuerdo
Ataque_Enemigo_En_X_Y_Z (identificador, x, y, z)	Visible(Enemigo)=Si Perceptible(Ghost)=Si Estado(PANCHO)=Conflicto Posición(Enemigo)=(x,y,z) Estado_Comunicación(Oficial)=ataque_enemigo_en

Tabla 5.11: Correspondencia entre Perceptos y Creencias

*La correspondencia que siguen perceptos: Descenso, Girar_derecha y Girar_izquierda, sigue un esquema similar.

5.8. Capacidades Reactivas del Agente

Basándonos en las acciones especificadas para el agente, así como en las expectativas vinculadas con dichas acciones, el comportamiento reactivo esperado en PANCHO responderá a un conjunto de reflejos particulares, especificador en esta sección.

Los actos reflejos vinculados a las acciones de naturaleza técnica para la presente concreción son los siguientes:

Ascender:

Disparadores:

- = (En_vuelo(Ghost), Sí)
- = (Estado_Comunicación(Oficial), ascender)

Justificadores:

- Ninguno

Acción Respuesta:

- ascender (cantidad)

Descender:

Disparadores:

- = (En_vuelo(Ghost), Sí)
- = (Estado_Comunicación(Oficial), descender)

Justificadores:

- Ninguno

Acción Respuesta:

- descender (cantidad)

Girar_Derecha:

Disparadores:

- = (En_vuelo(Ghost), Sí)
- = (Estado_Comunicación(Oficial), girar_derecha)

Justificadores:

- Ninguno

Acción Respuesta:

- girar_derecha(cantidad)

Girar_Izquierda:

Disparadores:

- = (En_vuelo(Ghost), Sí)
- = (Estado_Comunicación(Oficial), girar_izquierda)

Justificadores:

- Ninguno

Acción Respuesta:

- girar_izquierda (cantidad)

Eludir_enemigo:

Disparadores:

- = (Estado(Pancho), Peligro)

Justificadores:

- Ninguno

Acción Respuesta:

- eludir_enemigo()

Huir:

Disparadores:

- = (Estado(Pancho), Conflicto)

Justificadores:

- > (Miedo(Pancho), Umbral_Superior_Miedo(Pancho))

Acción Respuesta:

- huir()

5.9. Capacidades Sociales del Agente

En el capítulo 4 se ha definido que la interacción social del agente se dispara dentro del nivel social, es decir a través de metas, de forma análoga al nivel deliberativo. Sin embargo, en la presente concreción el diseño de la comunicación propuesta, no está estructurada en protocolos (como veremos en el ejemplo de la sección 6.2), sino que cada mensaje está planteado de una manera aislada, por tanto no pertenece de forma explícita a un esquema de intercambio de mensajes.

Por este motivo, se decidió utilizar capacidades reactivas para lanzar los mensajes, planteando un conjunto de disparadores (uno por cada mensaje) que a modo de reflejos, lancen las acciones de naturaleza social:

Situación en Vuelo?:

Disparadores:

- = (Estado(Pancho), Peligro)

Justificadores:

- > (Miedo(Pancho), Umbral_Superior_Miedo(Pancho))

Acción Respuesta:

- mensaje("Situación en vuelo?)

Situación en Tierra:

Disparadores:

- = (Estado(Pancho), Peligro)

Justificadores:

- > (Miedo(Pancho), Umbral_Superior_Miedo(Pancho))

Acción Respuesta:

- mensaje("Situación en tierra?")

Tiempo para arribo:

Disparadores:

- = (Estado(Pancho), Alerta)
- = (Estado_Comunicación(Oficial), tiempo_para_destino)

Justificadores:

- Ninguno

Acción Respuesta:

- mensaje(“Tiempo para arribo”, Minutos)

Ofensiva a criterio:

Disparadores:

- = (Estado_Comunicación(Oficial), permiso_para_ofensiva)
- = (Estado(Pancho), Conflicto)

Justificadores:

- < (Miedo(Pancho), Umbral_Superior_Miedo(Pancho)

Acción Respuesta:

- mensaje (“Ofensiva a criterio”)

Mantener la cordura:

Disparadores:

- = (Estado(Pancho), Peligro)

Justificadores:

- < (Miedo(Pancho), Umbral_Superior_Miedo(Pancho))

Acción Respuesta:

- mensaje(“Mantener la cordura”)

Trayecto recorrido (P %):

Disparadores:

- <= (Distancia(Zona_Segura) , 5 %)

Justificadores:

- > (Alegría(Pancho), Umbral_Superior_Alegría(Pancho))

Acción Respuesta:

- mensaje ("Trayecto recorrido", P %)

5.10. Evaluación de la Arquitectura

5.10.1. Comparativa de Esfuerzos

Para evaluar la cantidad de trabajo que ha supuesto el desarrollo de *Ghost Squadron* se registará el esfuerzo en la métrica *meses* × *persona*. Contando el equipo con tres desarrolladores de conocimientos y destrezas similares, esta medida puede dar una idea del esfuerzo requerido para elaborar la Concreción Contextual y, lo que es más importante, compararla con las medidas de esfuerzo (también en *meses x persona*) de las concreciones contextuales desarrolladas junto con la primera versión de COGNITIVA en los trabajos de [Leal, 2005, Molina, 2005]: *Sabana 3D* y *Subasta*s.

Para realizar esta comparativa, no se ha tenido en cuenta la fase de diseño de las concreciones, ya que en el caso de *Ghost Squadron* el diseño proviene fundamentalmente del escenario planteado en el trabajo de [Aguilar, 2008], y no se tienen datos del esfuerzo realizado por el autor. Por tanto, la comparativa se ciñe a la implementación de las contextualizaciones.

En la tabla 5.12 observamos la comparativa de esfuerzos entre las tres concreciones mencionadas, junto con la Concreción Funcional desarrollada en la primera versión de Cognitiva.

Lo primero que llama la atención es la desproporcionada relación entre el coste de desarrollar una contextualización para COGNITIVA (en otras palabras, un escenario) y una Concreción Funcional (que como se vio en 2.10.5, comprende la estructuras y procesos básicos de la arquitectura). Esta observación confirma la ventaja de COGNITIVA sobre otras arquitecturas al evitar la necesidad de rediseñarse cada vez que se construye un nuevo escenario.

En cuanto a la comparativa entre concreciones, se observa que la concreción *Ghost Squadron* pese a ser de un tamaño similar a *Sabana 3D* (como se puede observar en las tablas 5.13 y 5.14 de la siguiente sección) requiere un esfuerzo sustancialmente inferior, y se sitúa en unos valores de esfuerzo medios entre las dos concreciones originales.

El motivo de este inferior esfuerzo radica en el trabajo de rediseño explicado en el capítulo 3, que permite construir concreciones contextuales para COGNITIVA de una manera más sencilla. En la nueva versión de COGNITIVA se dan más responsabilidades a la Concreción Funcional,descargando así a la contextual, y se desacoplan totalmente, de forma que el desarrollador tiene mucho más claro qué ha de implementar y donde tiene que hacerlo.

5.10.2. Comparativa de Tamaños

Evaluar el tamaño de un producto software es, en las métricas habituales, hablar de número de líneas de código (LOC) y de número de clases desarrolladas. Sin embargo, el objetivo de establecer una comparativa de tamaños entre las Concreciones Contextuales desarrolladas hasta ahora sobre COGNITIVA, no puede basarse sólo en estos parámetros, debido a que el trabajo de rediseño y extensión de la arquitectura realizado en este proyecto ha transformado profundamente la manera de construir contextualizaciones, y la comparativa en estos términos no sería representativa.

De esta manera, pese a que se realizará una primera comparativa en función a los parámetros tradicionales, en la segunda parte de esta sección se muestra un análisis de la cantidad de elementos que se han identificado para cada uno de los componentes de la arquitectura:

Concreción	Esfuerzo (meses x persona)
Sabana 3D	0,38
Subastas	0,25
Ghost Squadron	0,33
Funcional (original)	10

Tabla 5.12: Comparativa de esfuerzos (en meses x persona) entre Concreciones

creencias, acciones, expectativas, eventos, perceptos, reflejos y reacciones.

Para determinar las métricas de número de Clases y LOC, se ha utilizado Eclipse Metrics, un plugin para el IDE Eclipse que permite determinar éstas y otras métricas de calidad para los proyectos desarrollados. Los resultados se pueden observar en la tabla 5.13.

Concreción	Clases	LOC
Sabana 3D	24	3025
Subastas	11	1114
Ghost Squadron	42	3108
Funcional (original)	72	4500
Funcional (nueva)	77	8307

Tabla 5.13: Tamaño en Clases y LOC de las Concreciones Contextuales

Analizando los resultados, la primera observación destacable es el importante aumento en LOC de la Concreción Funcional tras el rediseño y extensión realizado. El efecto de las nuevas capacidades agregadas a la arquitectura, más el aumento de responsabilidad otorgado a la parte Funciona, provoca que el tamaño en LOC de la nueva versión prácticamente duplique a la original.

A su vez, otro dato a destacar es el gran número de clases que comprende la nueva contextualización desarrollada, *Ghost Squadron*, en comparación con las ya desarrolladas. Esto se debe a que, tras el rediseño, cada acción de los agentes tiene una clase asociada (ver sección 3.1.2) y, en la primera versión, el código de las acciones se incluía todo junto en la clase Acción.

Finalmente es importante observar como, pese al aumento en número de clases, el tamaño en LOC de la nueva concreción se mantiene similar al de la concreción *Sabana 3D*. Este parecido en tamaños entre ambas concreciones, se confirma cuando acudimos a la comparativa de elementos de la arquitectura, en el cuadro 5.14.

Los resultados de la comparativa de tamaño en número de elementos nos deparan la confirmación de que *Ghost Squadron* es una contextualización semejante en tamaño a *Sabana 3D*. Esto nos anima a comparar ambas concreciones en términos de productividad para evaluar el rendimiento del equipo de desarrollo.

5.10.3. Comparativa de Productividad

En la tabla 5.15 se puede observar una comparativa entre la productividad del desarrollo de las contextualizaciones desarrolladas. Los parámetros de referencia son el número de

clases y el número de LOC entre el tiempo de desarrollo de cada concreción.

La primera conclusión a destacar es el aumento aparente de la productividad en clases/día que encontramos en el desarrollo de la concreción *Ghost Squadron*, bastante inesperado al tratarse de un equipo nuevo, sin experiencia en COGNITIVA. Sin embargo, rápidamente encontramos la justificación en el alto número de clases generadas en el escenario *Ghost Squadron* dado que con el nuevo diseño, se precisa de una clase por cada acción (ver sección 3.1.2).

Más razonables son los valores obtenidos en cuanto a productividad en LOC/día. El equipo se encuentra a medio camino entre los valores obtenidos por las otras dos concreciones, estando más próximo al de *Subastas*, como corresponde a un equipo nuevo, sin experiencia en la arquitectura. Por tanto los resultados son bastante buenos, puesto que mejora la productividad del equipo original de COGNITIVA cuando acometió su primer escenario, aunque queda bastante lejos de la excelente productividad conseguida por éste cuando desarrolló *Sabana 3D*, de tamaño similar a *Ghost Squadron.* Por tanto se confirma el importante efecto de la experiencia en el manejo de la arquitectura para obtener una alta productividad.

Constructor	Subastas	Sabana 3D	Ghost Squadron
Tipos de Individuos	1	2	1 (2 roles)
Creencias Totales	26	45	32
Rasgos de Personalidad	3	3	5
Emociones	2	5	3
Estados Físicos	3	1	0
Actitudes	0	6	0
Situación Actual	0	0	1
Acciones	3 (Concretas)	2 Abstractas y 11 Concretas	2 Abstractas y 10 Concretas
Expectativas	7	20	18
Intérprete	4 Eventos y 4 Perceptos	11 Eventos y 11 Perceptos	12 Eventos y 12 Perceptos
Comportamiento Reactivo	3	6	12

Tabla 5.14: Tamaño en Elementos de las Concreciones Contextuales

Concreción	Subastas	Sabana 3D	Ghost Squadron
Clases/día	2,2	3,2	3,8
LOC/día	222,5	403,33	282,54

Tabla 5.15: Productividad de las Concreciones Contextuales

Capítulo 6

OTROS CONTEXTOS DE CARÁCTER SOCIAL

6.1. Introducción

En este capítulo se detallará un ejemplo de aplicación del mecanismo de Interacción Social propuesto en el capítulo 4.

Se probará sobre la Concreción Contextual Sabana 3D que consiste en un entorno virtual que simula una sabana africana poblada por leones y cebras virtuales autónomos que realizan sus principales labores para asegurar su supervivencia. Este escenario se detalla en profundidad en [Cabeza, S.N.]. Para realizar las pruebas, únicamente contaremos con dos agentes *Cebra*, sin la presencia de ningún león. En este caso, el escenario planteado es el de un agente *Cebra* que le solicita a otro que beba una determinada cantidad de agua.

El ejemplo de aplicación se integra dentro del propósito del Nivel Social, planteado en COGNITIVA pero no desarrollado. Se trata, a grandes rasgos, de lograr que uno de los agentes pueda solicitar un servicio a otro agente, como una acción más dentro de un plan diseñado para satisfacer una meta. El propósito es probar su rendimiento y adecuación para integrarlo como la base del futuro Nivel Social (ver [Spinola, 2008]).

A diferencia de la concreción *Ghost Squadron*, aquí las interacciones no se limitan a un esquema de Acción/Reacción sino que tienen una cierta complejidad al seguir el protocolo de Solicitud de Servicio diseñado en la sección 4.3, con algunas particularizaciones.

6.2. Ejemplo de Interacción Social en Sabana 3D

En el futuro, este mecanismo de interacción estará integrado dentro de los planes de los agentes, por tanto para realizar las pruebas es preciso simular que la interacción tiene lugar en el transcurso de uno de estos planes.

Por ejemplo, la acción que da comienza el protocolo, Solicitar, debe ser lanzada a través de una reglaMeta, dado que, como la solicitud de la acción no está encuadrada dentro de un plan del nivel social, sino que es un plan en sí misma, debe forzarse el origen deliberativo/social de la solicitud para asemejar el ejemplo al estado en el que intervendrá normalmente el protocolo de solicitud, esto es, dentro de un proceso deliberativo/social. Así, la regla meta que dispara la solicitud es:

```
reglameta
tipo() = entero:1
distancia() < doble:6
estado_comunicacion() < cadena:finalizando_solicitud
finantecedente
prioridad:7
caducidad:100000
estado_comunicacion() = cadena:finalizando_solicitud
finmeta
```

Sin embargo, la acción que da respuesta en el agente servidor, Tratar Solicitud, debe ser lanzada a través de reglaMeta siempre, tanto en esta simulación, como en un caso completo de plan deliberativo. La explicación consiste en que, dado que la solicitud se encuadra dentro de un proceso deliberativo/social, la recepción y atención de dicha solicitud debe también proceder del nivel deliberativo/social. Uno de los motivos más importantes es que la prioridad de atención de este tipo de solicitudes debe ser baja, debido a que la prioridad de un agente debe ser, en primer lugar sus procesos reactivos, y en segundo lugar, sus propios procesos deliberativos.

La estructura de la reglaMeta que dispara el tratamiento de la solicitud es:

```
reglameta
estado_comunicacion() = cadena:analizando_solicitud
finantecedente
prioridad:6
caducidad:100000
estado_comunicacion() = cadena:inicial
finmeta
```

En el módulo conversor, al codificar las metas cuyo objetivo es un cambio en el estado de la comunicación, se establecen las respectivas metas *goal-solicitando* y *goal-sirviendo* que, una vez en el planificador, se descomponen en sendos planes que incluyen las acciones Solicitar y Tratar Solicitud, respectivamente.

Una vez definidos los disparadores del entorno, se procede a ejecutar la simulación, que sigue fielmente el guión previsto por el protocolo de Solicitud de Servicio, obteniendo la siguiente traza:

```
cebrita solicitar
cebrita enviarSolicitud
cebrita2 Ha llegado un REQUEST
cebrita2 tratarSolicitud
cebrita2 enviarConfirmacion
cebrita Ha llegado un AGREE
cebrita2 beber
cebrita tratarRespuesta
cebrita recibirResultado
cebrita2 tratarServicio
cebrita2 enviarResultado
cebrita2 terminarSolicitud
```

cebrita Ha llegado un INFORM

cebrita tratarResultado

cebrita terminarSolicitud

En la traza, se observa tanto las acciones realizadas por cada cebra, como los mensajes recibidos, de forma que se aprecia la correspondencia entre el flujo de mensajes y las acciones ejecutadas.

El tiempo de ejecución de todo el protocolo oscila entre los 18/20 segundos, por lo que su integración en el nivel social es perfectamente asumible: si la acción de un plan la realiza el propio agente, consumirá entre 1 y 2 segundos; mientras que si solicita su realización a otro, incrementa su latencia entre 10 y 20 veces. Esto, lejos de ser una desventaja, dota de gran realismo a la simulación e incrementa la verosimilitud de la misma, a la vez que refuerza la idea de un nivel social por encima del más prioritario y rápido nivel deliberativo.

A su vez, es preciso señalar que el protocolo es robusto frente a otras posibilidades, como:

- El caso en que la cebra que recibe la solicitud rechace realizar el servicio:

cebrita solicitar

cebrita enviarSolicitud

cebrita2 Ha llegado un REQUEST

cebrita2 tratarSolicitud

cebrita2 enviarRechazo

cebrita Ha llegado un REFUSE

cebrita2 terminarSolicitud

cebrita cancelar

cebrita terminarSolicitud

- El caso en que la cebra que recibe la solicitud no logre realizar el servicio:

cebrita solicitar

cebrita enviarSolicitud

cebrita2 Ha llegado un REQUEST

cebrita2 tratarSolicitud

cebrita2 enviarConfirmacion

cebrita Ha llegado un AGREE

cebrita tratarRespuesta

cebrita recibirResultado

cebrita2 tratarServicio

cebrita2 enviarFallo

cebrita2 terminarSolicitud

cebrita Ha llegado un FAILURE

cebrita tratarResultado

cebrita cancelar

cebrita terminarSolicitud

Capítulo 7

CONCLUSIONES

En este punto recopilaremos todas las conclusiones obtenidas tras el trabajo realizado a lo largo de este Proyecto Fin de Carrera, haciendo especial hincapié en las técnicas y metodologías utilizadas. De cualquier modo, hay que destacar que se ha finalizado con éxito la actualización y mejora de la arquitectura COGNITIVA, así como la extensión de nuevas capacidades.

Es preciso señalar que se han logrado sentar las bases del nivel social, tras incorporar el mecanismo de interacción entre agentes. Además, al desarrollar las extensiones y mejoras de la arquitectura, se han aprendido nuevas tecnologías de programación y afianzado conocimientos de ingeniería del software.

Dividiremos el capítulo en:

- Conclusiones acerca de las metodologías de desarrollo.
- Conclusiones sobre los lenguajes y herramientas utilizadas.
- Conclusiones acerca de la arquitectura COGNITIVA.

7.1. Metodologías de Desarrollo

Tanto para la parte de rediseño y extensión de la arquitectura (ver capítulo 3) como para la interacción social (ver capítulo 4), las metodologías de desarrollo fundamentalmente utilizadas fueron: Gaia [Zambonelli et al., 2003], para cubrir el desarrollo de los agentes, y el Proceso Unificado [Jacobson et al., 2000], para lo referente al diseño de los objetos.

Dado que el núcleo de la arquitectura, así como los niveles reactivo y deliberativo, estaban ya desarrollados cuando comenzó este proyecto, no ha sido necesario aplicar Gaia en los niveles de Análisis ni de Diseño Arquitectónico puesto que los artefactos producidos en estas fases no se veían alterados por las extensiones y mejoras realizadas. No obstante, fue necesario hacer uso de Gaia tanto en la fase de rediseño y extensión (ver capítulo 3), puesto que se modificaron algunos comportamientos de los agentes, como a la hora de introducir la Interacción Social (ver capítulo 4)

Sin embargo, es precisamente al utilizar el nivel de Diseño Detallado cuando encontramos a Gaia bastante limitada para facilitar el paso a la implementación. Una vez se han definido los agentes (Modelo de Agentes), y los servicios ofrecidos por éstos (Modelo de Servicios), la metodología deja de ofrecer modelos de más bajo nivel que permitan pasar a la implementación fácilmente. Concluimos que Gaia es una metodología poco útil para realizar modificaciones y ampliaciones sobre sistemas multiagente ya desarrollados, puesto que no proporciona artefactos que modelicen los conceptos más cercanos a la implementación, como son los protocolos y los comportamientos.

Por estas razones, el uso de Gaia en el presente proyecto ha sido bastante limitado. Como ya se ha señalado en la sección 3.1.3, ha sido fundamental acudir a Extensiones sobre el Diseño Detallado de Gaia para poder especificar las modificaciones y extensiones realizadas sobre la arquitectura, puesto que es allí donde encontramos modelos como los Protocolos Detallados que permiten especificar los cambios en los mensajes intercambiados entre los agentes.

Avanzando un poco más en la experiencia acerca de la metodología utilizada en el desarrollo de los agentes, podemos concluir que el análisis basado en roles que propone la metodología Gaia no es el más indicado para la construcción de nuevas concreciones contextuales de COGNITIVA. En realidad, para el análisis de una nueva concreción, los artefactos que ofrece la propia arquitectura, como son la especificación de las Creencias de los agentes y la formalización de sus Acciones, permite, junto con las Extensiones sobre el Diseño Detallado de Gaia mencionado, afrontar la implementación del entorno con calidad y fiabilidad, como demuestra el éxito del desarrollo de la concreción *Ghost Squadron* (ver capítulo 5).

Por otro lado, para la fase de rediseño y extensión se aplicó el Proceso Unificado. Los Diagramas de Clase han sido la herramienta básica para aplicar las técnicas de refactorización propuestas, que obedecían a las recomendaciones descritas acerca de temas como el uso de jerarquías y el reparto de responsabilidades. Se ha tomado la decisión de no obtener artefactos acerca de los Casos de Uso del sistema, pese a ser uno de los fundamentos del

Proceso Unificado, debido a que esa aproximación corresponde más a las fases de Análisis y Diseño de Alto Nivel que ya quedan cubiertas con los Modelos de Gaia para roles y servicios.

Otra posibilidad hubiera sido utilizar el Diseño de Agentes, otra Extensión del Diseño Detallado de Gaia, para esta fase, dado que ayuda a obtener el diseño de clases de cada agente. Sin embargo se optó por los Diagramas de Clase del Proceso Unificado debido a que todos los agentes presentaban una estructura de clases común, además de ser ésta considerablemente más compleja que la de los agentes de un sistema multiagente común.

COGNITIVA propone una arquitectura interna de agente con muchos componentes y estructuras diferentes, relacionados de una forma más o menos débil (según el caso). Por ello se ha optado por construir diagramas de clase parciales en lugar de uno completo, de forma que resultara práctico trabajar sobre ellos y aportaran verdadera información al desarrollador.

7.2. Lenguajes y Herramientas

La versión inicial de COGNITIVA fue desarrollada con la plataforma JADE, y debido a los buenos resultados obtenidos originalmente, se ha continuado utilizando en este nuevo desarrollo, actualizando sin embargo a la nueva versión de la plataforma.

Una vez más, la experiencia con JADE ha resultado muy satisfactoria debido a que, al ser una ampliación de Java para desarrollar agentes, resulta tan sencillo implementar sobre ella como hacerlo en Java. Además existe una estrecha relación entre los modelos de la Extensión sobre el Diseño Detallado de Gaia utilizados (ver definición en la sección 3.1.3) y la implementación en Jade.

Aun así existen dos factores que complican y ralentizan el desarrollo en JADE, y que deberán ser solucionados en los futuros lenguajes orientados a agentes. El primero es la lenta ejecución de cada simulación debido a la gran cantidad de protocolos internos que se ponen en marcha para comunicar y gestionar los agentes; esto convierte la depuración en una tarea aún más ardua que en otros lenguajes.

El segundo es la propia depuración en sí ya que, pese a contar con una herramienta gráfica para controlar la interacción entre los agentes, es complicado seguir el foco de ejecución en cada momento, al estar cada comportamiento de un agente ejecutado sobre un *thread* distinto.

Otra enseñanza adquirida ha sido a través del uso del algoritmo JSHOP2 (ver sección 3.6) y del framework Jena (ver sección3.8), que utilizan lenguajes declarativos, cercanos a la lógica, para algunas tareas como traducir los planes al planificador y realizar consultas elaboradas sobre la ontología. Ha supuesto todo un reto, superado con éxito gracias a la documentación existente y la ayuda de algunos compañeros del Laboratorio Decoroso Crespo. Además nos ha permitido combinar por un tiempo el desarrollo de agentes, que se ha nutrido fundamentalmente de nuestro aprendizaje en Ingeniería del Software y Programación Imperativa, con la aplicación de otro tipo de conocimientos adquiridos en asignaturas como Ingeniería del Conocimiento y Programación Lógica. Nos permite además concluir que el desarrollo de agentes se basa en elementos tanto de las disciplinas de Ingeniería del Software como de la Inteligencia Artificial.

Por otro lado, en cuanto a herramientas, utilizar el IDE Eclipse ha proporcionado muy buenos resultados gracias a ofrecer muchos recursos junto a una velocidad más que aceptable, gracias a su sistema de *plugins* para incorporar al entorno las herramientas necesarias en cada momento.

Creemos que el uso de herramientas *Open Source* ha ayudado de una forma decisiva a lograr concluir con éxito el proyecto. Desde el propio Eclipse, en el que siempre encontramos cualquier *plugin* necesario ya desarrollado por algún miembro de la comunidad, hasta los framework de Jade y Jena, los cuales aprendimos a manejar gracias a la abundancia de documentación de la comunidad sobre su API y funcionamiento interno.

Son a su vez *libres* los envoltorios desarrollados por nuestros compañeros del laboratorio para JSHOP2 y Jena (ver [Maján, 2009, García de la Torre, S.N.]) cuyo uso nos facilitó implantar ambas plataformas en la arquitectura, a la vez que comprendíamos su funcionamiento observando el código de los envoltorios. Inclusive se llegó a plantear la necesidad de realizar algunas modificaciones sobre el algoritmo de planificación de JSHOP2, que finalmente no fueron necesarias gracias a la modificación de algunas partes del envoltorio (ver [Maján, 2009]), pero que a su vez hubieran sido posibles dado que JSHOP2 también es *software libre*.

Finalmente, una constante a lo largo de todo el proyecto, ha sido la sensación de estar utilizando tecnologías demasiado limitadas al ámbito académico y que rara vez se aplican en el mundo empresarial. Desde la orientación a agentes, que es utilizada aún de forma marginal en proyectos comerciales, hasta la utilización de herramientas que, si bien tienen la ventaja de ser *Open Source*, tan sólo cuentan con respaldo universitario, como Protégé (Stanford).

La excepción, y el camino a seguir para el *software libre*, la conforman los framework Jade

y Jena que cuentan con el soporte de Telecom Italia y Hewlett-Packard respectivamente.

7.3. Arquitectura COGNITIVA

La aportación de este proyecto sobre COGNITIVA se ha dividido en tres grandes fases de las que podemos sacar conclusiones de forma separada.

Rediseño y Extensión Esta fase ha sido la que más tiempo ha requerido, debido a que abarca todo el análisis de lo realizado originalmente sobre COGNITIVA, junto con el desarrollo de todas las mejoras y actualizaciones sobre distintos aspectos de la arquitectura, a excepción del nivel social.

En conjunto podemos señalar que la principal lección aprendida es la importancia de un buen diseño para que programadores distintos a los que desarrollaron inicialmente un sistema, puedan comprenderlo y aportar nuevas funcionalidades al mismo de una manera relativamente sencilla. Ha sido necesario un exhaustivo proceso de refactorización del código para adecuar el diseño encontrado a las normas y especificaciones que se establecen en el Proceso Unificado, y concretamente, a los Patrones de diseño.

No obstante, la versión inicial de COGNITIVA fue desarrollada en un tiempo muy ajustado, por lo que se decidió dar preferencia a asegurar el desarrollo de todas las funcionalidades antes que a la calidad del código implementado.

Además, se ha comprendido que es crucial facilitar al máximo el desarrollo de nuevas contextualizaciones, tratando de desacoplarlo de cuestiones funcionales y acercándolo al lenguaje natural.

Por último, se han descubierto nuevas funcionalidades para elementos ya definidos en la arquitectura, como las expectativas, las acciones,o las creencias sobre la situación actual, que confirman la buenas líneas apuntadas en [Imbert, 2005].

Interacción Social En la segunda fase se incorporó el mecanismo de Interacción Social en un tiempo bastante rápido en función de las previsiones realizadas.

La principal lección obtenida, apuntada ya en [Spinola, 2008], fue que la mayor parte de los mecanismos establecidos para el nivel deliberativo son análogos (o los mismos) que los del nivel social.

Además, dado que el objetivo en esta aproximación era establecer las bases de la Comunicación, se ha descubierto que hay bastante trabajo ya realizado, gracias a los modelos de protocolos estándar de FIPA ACL. El desarrollo del nuevo protocolo de Solicitud de Servicio para el nivel social ha resultado ser muy similar al FIPA Request Interaction Protocol Specification definido por FIPA [FIPA Request, 2002], residiendo la mayor complejidad en la encapsulación de los mensajes en acciones COGNITIVA.

Finalmente, se ha vuelto a concluir que COGNITIVA es una arquitectura con grandes posibilidades de ampliación al encontrar, una vez más, una nueva funcionalidad para las expectativas para asegurar la sincronización en la comunicación entre los agentes.

Concreción Contextual Ghost Squadron En esta tercera y última fase, el desarrollo de una nueva contextualización, se han obtenido unos importantes resultados en esfuerzo y productividad (ver sección 5.10).

Una vez más se concluye con la importancia de dividir el desarrollo en dos concreciones, funcional y contextual, de forma que cada vez que se necesite desarrollar un nuevo entorno de simulación (es decir, un ejemplo) tan sólo sea preciso desarrollar los componentes de la Concreción Contextual, obteniendo un importantísimo ahorro en tiempo y esfuerzo de comprensión. En la sección 5.10.1 observamos como la proporción del esfuerzo necesario para desarrollar una Concreción Funcional representaba hasta treinta veces el requerido para una nueva Contextualización.

Además, se ha comprobado como el nuevo diseño facilita el desarrollo de nuevas contextualizaciones, reduciendo el esfuerzo, y permitiendo aumentar la productividad del equipo de desarrollo, aún siendo un equipo sin experiencia en COGNITIVA.

Las pruebas realizadas en este entorno sobre el mecanismo de Interacción Social proporcionan la conclusión de que el componente fundamental a garantizar es la coherencia del estado de los agentes durante la comunicación y al final de la misma. Además, es crucial gestionar la concurrencia de todo el sistema.

Por último, respecto a la experiencia de los usuarios en el entorno desarrollado, hemos de concluir que la ausencia de un entorno 3D ha pesado en la capacidad de los usuarios para determinar el modelo personal de los agentes en función de su comportamiento. Sin embargo, la interfaz desarrollada ha sido suficiente para poder llevar a cabo las pruebas del trabajo de [Aguilar, 2008] sobre Estrategias de Entrenamiento Colaborativo y obtener conclusiones de gran valor.

Capítulo 8

LÍNEAS DE TRABAJO FUTURO

Este proyecto está englobado dentro de una tarea mucho más grande, que consiste en completar la arquitectura COGNITIVA. Por tanto, igual que no se partió de cero, sino que se comenzó a partir de los trabajos de [Leal, 2005, Molina, 2005], se continuará el desarrollo aquí presentado, incorporando nuevas características y funcionalidades a la arquitectura. Estas nuevas líneas de trabajo han ido apareciendo según se avanzaba en el proyecto, en algunas ocasiones al echar en falta una nueva capacidad, y en otras al establecer los límites del presente proyecto. Seguidamente se describen las más relevantes:

Nivel Social definitivo En este proyecto se han sentado las bases del nivel social, incorporando las capacidades de interacción entre agentes y los protocolos de servicio. Sin embargo nada se ha dicho de cómo incorporar estos mecanismos al comportamiento del agente, esto es, cómo el agente decide qué necesita solicitar, cómo establece a qué otro agente solicitarlo, y otras características más que conforman realmente el nivel social, de manera que se defina con un detalle similar al Deliberativo.

Para esta tarea se hace preciso el estudio de las normas y las dinámicas sociales, y es la línea de trabajo en que se basa la investigación de [Spinola, 2008].

Nuevos Protocolos En la sección 4.2.4.11 se ha diseñado el protocolo de Solicitud de Servicio mediante el cuál, un agente solicita otro la realización de una acción. Este protocolo admite que el agente servidor acepte o rechace la realización de la acción. Como una futura mejora, se podrían añadir nuevos protocolos como, por ejemplo, un protocolo de negociación en el que los agentes aceptaran llevar a cabo el servicio a cambio de otro, o a partir de unos parámetros negociados (tiempo, calidad, ...).

Siguiendo en esta línea, se podrían diseñar protocolos más complejos, como la cooperación entre dos o más agentes, que se asocian para llevar a cabo acciones beneficiosas para todos, o la coordinación entre un grupo, para que una parte de ellos se encargue de una tarea, mientras el resto de otra distinta.

Inferencias Ontológicas Se ha señalado que la otra gran extensión realizada en este desarrollo, es la inclusión de una Ontología para el manejo de las Creencias del agente. Todo lo referente a la misma se explica en [Cabeza, S.N.] y, a grandes rasgos, se fundamenta en sustituir el sistema de Listas Enlazadas por una Ontología basada en Jena. Sin embargo, el desarrollo realizado sólo abarca funcionalidades de inserción, consulta simple, modificación y eliminación de predicados; queda pendiente como nueva ampliación explotar el mecanismo de inferencias que permite la Ontología.

Gracias al mismo, se podrían diseñar reglas que deduzcan información que no aparece de forma explícita en la base de conocimiento, pero que se infiere a partir de un conjunto de predicados relacionados.

Además, el abanico de tipos de conocimiento que el agente puede manejar, se amplía enormemente. Hasta ahora, se veía limitado por la estructura rígida de las creencias: objeto-atributo-valor. Sin embargo, gracias al sistema de predicados, se pueden definir sub creencias o atributos jerárquicos (un agente *cebra* será a la vez *animal*).

Interfaz Ficheros Inicialización Agente Una de las mejoras más importantes de cara a los futuros desarrolladores de entornos virtuales basados en COGNITIVA, es el nuevo formato de los ficheros de configuración para inicializar los agentes. En síntesis, se ha sustituido un lenguaje de basado en códigos numéricos, por otro más cercano al natural, y que está bastante cerca de los formatos de etiquetas basados en XML.

No obstante, lo más deseable para una rápida configuración de los agentes (fundamental a la hora de realizar distintas simulaciones variando los parámetros), es contar con una interfaz gráfica que, a modo de ecualizador visual, establezca los valores de cada creencia y elemento del modelo personal.

Historia Pasada En este proyecto se ha tratado de extender y actualizar todos los componentes de la arquitectura propuestos en [Imbert, 2005]. Sin embargo, el mecanismo de Historia Pasada no se ha desarrollado, puesto que no era crítico para el resto de extensiones y concreciones contextuales desarrollados. En cualquier caso, es ya una tarea

ineludible, puesto que para refinar el nivel social se hace preciso una estructura donde recoger los eventos e interacciones sucedidas, de cara a tomar decisiones sobre el resto de agentes y conversar con ellos de manera coherente con los hechos acaecidos.

Al desarrollar el Entorno Virtual Ghost Squadron (ver capítulo 5) se pedía el requisito de que los agentes mantuvieran el último mensaje recibido desde cada uno de los otros agentes. Esta necesidad, que parecía requerir de la utilización de la Historia Pasada, fue cubierta gracias a las creencias sobre la situación actual (ver sección 3.5).

Organizador Análogamente a lo ocurrido con la Historia Pasada, en este desarrollo no consideró la sustitución del Organizador actual (una cola de dos niveles, en la que el primer nivel es la prioridad, y el segundo las acciones de cada prioridad), por otra estructura más compleja. Sin embargo, la incorporación definitiva del nivel social plantea la necesidad de permitir ejecutar varias acciones concurrentemente. Para dotar de verosimilitud al agente, se hace preciso que éste pueda comunicarse con otro, mientras realiza alguna tarea, máxime si ésta es de larga duración en el tiempo.

Una propuesta sencilla, pero que permitiría una cierta concurrencia, es añadir un nuevo nivel a la cola. Con un nivel de capas (reactiva, deliberativa y social) se podría actuar de forma semi-concurrente, estableciendo unos ciertos mecanismos que garantizaran que ninguno de los niveles provoca estados incoherentes en otro.

Entorno Gráfico 3D En la primera versión de COGNITIVA se contó con un entorno virtual 3D para el mundo Sabana, gracias al prototipo desarrollado en el Laboratorio Decoroso Crespo dentro del proyecto MAEVIF(ver [de Antonio et al., 2005]) . Ahora este prototipo ha sido mejorado, y debido a la escasez de tiempo, se optó por no adaptar el Actuador y el Perceptor a la nueva tecnología.

Así, en el presente trabajo, las simulaciones y pruebas realizadas para la Concreción Contextual Ghost Squadron no se hicieron sobre el entorno virtual, sino que se optó por una rápida interfaz modo texto, con algunos elementos gráficos en 2D, basada en Java Swing. Sin embargo, se asume que no contar con un entorno 3D sobre el que desarrollar nuevas contextualizaciones es una limitación demasiado grande, por lo que es una línea prioritaria de desarrollo el adaptar la arquitectura COGNITIVA al nuevo modelo desarrollado en el Laboratorio.

Bibliografía

[Imbert, 2005] Imbert, Ricardo (2005) *Una Arquitectura Cognitiva Multinivel para Agentes con Comportamiento Influido por Características Individuales y Emociones, Propias y de Otros Agentes.* Tesis Doctoral. Facultad de Informática, UPM.

[Cabeza, S.N.] Cabeza, Alberto [S.N], [S.L].

[Leal, 2005] Leal, Pablo (2005) *Proyecto Cognitiva: Arquitectura Cognitiva en su Nivel Reactivo.* TFC. Facultad de Informática, UPM.

[Molina, 2005] Molina, Javier (2005) *Proyecto Cognitiva: Arquitectura Cognitiva en su Nivel Deliberativo.* TFC. Facultad de Informática, UPM.

[Ilghami., 2006] Ilghami , Okhtay (2006) *Documentation for JSHOP2*. Technical Report CS-TR-4694. Department of Computer Science, University of Maryland.

[Maján, 2009] Maján Álvaro (2009) *Integración del Planificador JSHOP2 en un Entorno Virtual Inteligente.* TFC. Facultad de Informática, UPM

[García de la Torre, S.N.] García de la Torre, David [S.N], [S.L].

[Aguilar, 2008] Aguilar, Raúl Antonio (2008) *Entrenamiento de grupos: Una Estrategia Asistida por Entornos Virtuales Inteligentes.* Tesis Doctoral. Facultad de Informática, UPM.

[Belbin, 1981, 1993] Belbin, M (1981, 1993) *Management Teams, Team Roles at work.* Butterworth Heinemann. Oxford.

[Costa and McCrae, 1985] Costa, P.T., & McCrae, R.R. (1985) *The NEO personality inventory manual.* Psychological Assessment Resources. Odessa, FL.

[Digman and Inouye, 1986] Digman, J. M., & Inouye, J. (1986) *Further specification of the five robust factors of personality.* Journal of Personality and Social Psychology, 50, 116-123.

[Spinola, 2008] Spinola, Jackeline (2008) *Especificación del Nivel Social de una Arquitectura Cognitiva para Agentes Cooperativos.* Tesis de Máster. Facultad de Informática, UPM.

[DRAE, 2001] Real Academia Española (2001) *Diccionario de la Real Academia de la Lengua Española, 22ª Edición.* Real Academia Española. Madrid

[Austin, 1962] Austin, J.L. (1962) *How to Do Things with Words.* J. O. Urmson. Oxford.

[Cortés, 2006] Cortés, Ulises (2006) *Comunicación entre Agentes Autónomos.* http://www.lsi.upc.es/~ia/ComunicacionA2006.ppt (11/04/2009).

[Weiss, 2000] Weiss, G (2000) *Multiagent Systems.* The MIT Press. Cambridge.

[Roman, 2004] Roman, Dumitru (2004) *ACL Agents Comunication Languages.* http://www.wsmo.org/papers/presentations/ACLs.ppt (11/04/2009).

[Botía, 2005] Botía, Juan A. (2005) *The FIPA Standard.* http://ants.dif.um.es/~juanbot/page_files/escuelaAgentes2005fipa.pdf (11/04/2009).

[de Antonio et al., 2005] A. de Antonio, J. Ramírez, R. Imbert y G. Méndez (2005) *Intelligent Virtual Environments for Training: An Agent Based Approach*. En M. Pěchouček, P. Petta y L. Z. Varga (eds.) Multi-Agent Systems and Applications IV. Lecture Notes in Artificial Intelligence, vol. 3690, págs. 82-91. Springer.

[Jacobson et al., 2000] Ivar Jacobson, Grady Booch, James Rumbaugh (2000) *El Proceso Unificado de Desarrollo de Software.* Pearson Addisson-Wesley. Madrid.

[Zambonelli et al., 2003] Franco Zambonelli, Nicholas R. Jennings, Michael Wooldridge (2003) *Developing Multiagent Systems: The Gaia Methodology.*

[FIPA] FIPA *Foundation for Intelligent Physical Agents*. http://www.fipa.org (11/04/2009).

[KQML] KQML *Knowledge Query and Manipulation Language.* http://www.cs.umbc.edu/kqml/ (11/04/2009).

[KIF] KIF *Knowledge Interchange Format.* http://logic.stanford.edu/kif/kif.html (11/04/2009).

[Galeano, 1989] Galeano, Ernesto César (1989) *Modelos de Comunicación.* Macchi. Buenos Aires, Argentina.

[FIPA Request, 2002] FIPA Request Interaction Protocol Specification SC00026H *http://www.fipa.org/specs/fipa00026/index.html* (11/04/2009).

[Odell et al., 2001] James J. Odell, H. Van Dyke Parunak, Bernhard Bauer, Agent-Oriented Software Engineering, Paolo Ciancarini and Michael Wooldridge (2001) *Representing Agent Interaction Protocols in UML.* Springer. Berlin, pp. 121-140, 2001.

[Imbert, 2006] Ricardo Imbert (2006) *Documentación asignatura Profundización en Ingeniería del Software.* http://is.ls.fi.upm.es/docencia/profundizacion/documentacion.html (11/04/2009).

Apéndice A

FICHEROS DE CONFIGURACIÓN GHOST SQUADRON

En este primer apéndice, se muestran los ficheros de configuración del agente *Piloto* de la Concreción Contextual Ghost Squadron desarrollada en el capítulo 5. Estos ficheros son los determinados en la sección 3.2.

Se incluyen en el proyecto para mostrar la información de entrada recibida por los agentes COGNITIVA, así como facilitar la comprensión del paso desde el diseño de una Concreción Contextual a su implementación funcional. Además, permite ver un ejemplo real de las mejoras introducidas en el formato de los ficheros (plantillas mostradas en la sección 3.2) para hacerlas más intuitivas para el usuario.

A.1. Fichero de Creencias

Personalidad

extraversion(yo) difuso:ligeramente

amabilidad(yo) difuso:ligeramente

responsabilidad(yo) difuso:ligeramente

estabilidad_emocional(yo) difuso:ligeramente

apertura(yo) difuso:ligeramente

Emociones

miedo(yo) difuso:ligeramente degradado:1.0002

alegria(yo) difuso:ligeramente degradado:0.009

sorpresa(yo) difuso:ligeramente degradado:0.0009

Estados Fisicos

Actitudes

Creencias

posicion_x(ghost) doble:0

posicion_y(ghost) doble:30

posicion_z(ghost) doble:1000

limiteY0(zona_segura) doble:0

limiteY1(zona_segura) doble:60

limiteX0(zona_segura) doble:120

limiteX1(zona_segura) doble:180

limiteZ0(zona_segura) doble:0

limiteZ1(zona_segura) doble:12000

limiteY0(zona_enemiga) doble:0

limiteY1(zona_enemiga) doble:60

limiteX0(zona_enemiga) doble:0

limiteX1(zona_enemiga) doble:120

limiteZ0(zona_enemiga) doble:0

limiteZ1(zona_enemiga) doble:12000

distancia(zona_segura_origen) doble:0

distancia(zona_segura_destino) doble:120

en_vuelo(ghost) entero:1

vista(enemigo) doble:12

estado_interno(yo) cadena:Alerta

rol_equipo(yo) cadena:Coordinador

Intereses

miedo difuso superior:bastante inferior:nada

alegria difuso superior:absolutamente inferior:ligeramente

sorpresa difuso superior:medianamente inferior:nada

RelacionesCreencias

influyente:extraversion influida:alegria acentua difuso:bastante

influyente:extraversion influida:sorpresa acentua difuso:medianamente

influyente:responsabilidad influida:miedo bipolar_mas difuso:bastante

influyente:estabilidad_emocional influida:miedo bipolar_menos difuso:medianamente

influyente:apertura influida:sorpresa atenua difuso:medianamente

PersoActitudes

ActitudEmociones

CreenciasIntereses

influyente:responsabilidad influida:miedo acentua superior difuso:ligeramente

influyente:estabilidad_emocional influida:miedo acentua superior difuso:medianamente

influyente:apertura influida:sorpresa acentua superior difuso:medianamente

influyente:extraversion influida:alegria atenua inferior difuso:ligeramente

EstadoEmociones

Estas son las creencias para el rol de equipo *Coordinador*, para el que el agente se comportara como un *Implementador* bastaría sustituir la línea:

rol_equipo(yo) cadena:Coordinador

por esta:

rol_equipo(yo) cadena:Implementador.

A.2. Fichero de Acciones

ascender

descender

girarDerecha

girarIzquierda

situacionEnTierra

situacionEnVuelo

tiempoParaArribo

ofensivaACriterio

mantenerCordura

trayectoRecorrido

deAcuerdo

enDesacuerdo

derribado

derribar

seguro

eludirEnemigo

huir

A.3. Fichero de Reglas

regla

en_vuelo(ghost) = entero:1

estado_comunicacion() = cadena:ascender

posicion_z(ghost) < doble:11001

finantecedente

prioridad:1

accion:ascender

Parametros

Parametro doble:1000

finregla

regla

en_vuelo(ghost) = entero:1

posicion_z(ghost) > doble:1999

estado_comunicacion() = cadena:descender

finantecedente

prioridad:1

accion:descender

Parametros

Parametro doble:1000

finregla

regla

en_vuelo(ghost) = entero:1

estado_comunicacion() = cadena:gira_izquierda

finantecedente

prioridad:1

accion:girarIzquierda

Parametros

Parametro doble:3

finregla

regla

en_vuelo(ghost) = entero:1

estado_comunicacion() = cadena:gira_derecha

finantecedente

prioridad:1

accion:girarDerecha

Parametros

Parametro doble:3

finregla

regla

estado_interno(yo) = cadena:Seguro

finantecedente

prioridad:1

accion:seguro

Parametros

finregla

regla

estado_comunicacion() = cadena:derribado

finantecedente

prioridad:1

accion:derribado

Parametros

finregla

regla

en_vuelo(ghost) = entero:1

estado_comunicacion() = cadena:tiempo_para_destino

finantecedente

prioridad:1

accion:tiempoParaArribo

Parametros

```
finregla
regla
posicion_x(ghost) > doble:113
rol_equipo(yo) = cadena:Coordinador
finantecedente
prioridad:1
accion:trayectoRecorrido
Parametros
finregla
regla
estado_interno(yo) = cadena:Peligro
rol_equipo(yo) = cadena:Coordinador
finantecedente
prioridad:1
accion:mantenerCordura
Parametros
finregla
regla
estado_interno(yo) = cadena:Peligro
rol_equipo(yo) = cadena:Coordinador
finantecedente
prioridad:1
accion:eludirEnemigo
Parametros
finregla
regla
estado_interno(yo) = cadena:Conflicto
```

finantecedente

prioridad:1

accion:huir

Parametros

finregla

regla

en_vuelo(ghost) = entero:1

estado_comunicacion() = cadena:derribar

finantecedente

prioridad:1

accion:derribar

Parametros

finregla

regla

en_vuelo(ghost) = entero:1

estado_comunicacion() = cadena:permiso_para_ofensiva

estado_interno(yo) = cadena:Conflicto

finantecedente

prioridad:1

accion:ofensivaACriterio

Parametros

finregla

regla

en_vuelo(ghost) = entero:1

estado_interno(yo) = cadena:Peligro

finantecedente

prioridad:1

```
accion:situacionEnTierra
Parametros
finregla
regla
en_vuelo(ghost) = entero:1
estado_interno(yo) = cadena:Peligro
finantecedente
prioridad:1
accion:situacionEnVuelo
Parametros
finregla
```

Apéndice B

FICHEROS DE CONFIGURACIÓN SABANA 3D

En este siguiente apéndice, se muestran los ficheros de configuración de los agentes *Cebrita Cliente* y *Cebrita Servidor* de la Concreción Contextual Sabana 3D desarrollada en el capítulo 6.

El objetivo de este apéndice es mostrar como se incluyen las nuevas capacidades de Interacción Social dentro de los ficheros de configuración originales de la arquitectura. Como ejemplo, podemos destacar las creencias sobre el *Estado de la Comunicación* incluidas en el fichero de Creencias, así como las reglas que lanzan las metas sociales (definidas sobre el *Estado de la Comunicación)* incluidas en el fichero de Reglas Metas.

B.1. Ficheros de Cebrita Cliente

B.1.1. Fichero de Creencias

```
Personalidad
valor(yo) difuso:ligeramente
fortaleza(yo) difuso:ligeramente
Emociones
miedo(yo) difuso:nada degradado:0.01
```

alegria(yo) difuso:medianamente degradado:0.008

sorpresa(yo) difuso:medianamente degradado:0.02

Estados Fisicos

sed(yo) valor difuso:bastante reposo difuso:absolutamente degradado:0.0003

hambre(yo) valor difuso:nada reposo difuso:absolutamente degradado:0.0003

cansancio(yo) valor difuso:nada reposo difuso:nada degradado:0.005

Actitudes

Creencias

limiteX0(sabana) doble:0

limiteX1(sabana) doble:100

limiteY0(sabana) doble:0

limiteY1(sabana) doble:100

distancia_minima(yo) doble:6

vista(yo) doble:10

estado_interno(yo) cadena:inicial

Intereses

miedo difuso superior:ligeramente inferior:nada

alegria difuso superior:absolutamente inferior:ligeramente

sorpresa difuso superior:absolutamente inferior:nada

sed difuso superior:medianamente inferior:ligeramente

hambre difuso superior:medianamente inferior:ligeramente

cansancio difuso superior:medianamente inferior:nada

RelacionesCreencias

influyente:valor influida:miedo disminuye difuso:ligeramente

influyente:valor influida:sorpresa disminuye difuso:ligeramente

PersoActitudes

influyente:valor influida:temor disminuye difuso:ligeramente

ActitudEmociones

influyente:temor influida:miedo aumenta difuso:bastante

influyente:temor influida:alegria disminuye difuso:medianamente

influyente:temor influida:sorpresa aumenta difuso:ligeramente

CreenciasIntereses

influyente:valor influida:miedo aumenta superior difuso:ligeramente

influyente:fortaleza influida:miedo aumenta superior difuso:ligeramente

influyente:fortaleza influida:sed aumenta superior difuso:ligeramente

influyente:fortaleza influida:hambre aumenta superior difuso:ligeramente

influyente:fortaleza influida:cansancio aumenta superior difuso:ligeramente

EstadoEmociones

influyente:sed influida:alegria disminuye difuso:medianamente

influyente:hambre influida:alegria disminuye difuso:medianamente

influyente:cansancio influida:alegria disminuye difuso:medianamente

B.1.2. Fichero de Acciones

cancelar

enviarSolicitud

recibirResultado

solicitar

terminaSolicitud

tratarRespuesta

tratarResultado

B.1.3. Fichero de Reglas Metas

```
reglameta
tipo() = entero:1
distancia() < doble:6
estado_comunicacion() < cadena:finalizando_solicitud
finantecedente
prioridad:7
caducidad:1000000
estado_comunicacion() = cadena:finalizando_solicitud
finmeta
```

B.2. Ficheros de Cebrita Servidor

B.2.1. Fichero de Creencias

```
Personalidad
valor(yo) difuso:medianamente
fortaleza(yo) difuso:medianamente
Emociones
miedo(yo) difuso:nada degradado:0.001
alegria(yo) difuso:medianamente degradado:0.008
sorpresa(yo) difuso:medianamente degradado:0.02
Estados Fisicos
sed(yo) valor difuso:bastante reposo difuso:absolutamente degradado:0.0003
hambre(yo) valor difuso:nada reposo difuso:absolutamente degradado:0.0003
cansancio(yo) valor difuso:nada reposo difuso:nada degradado:0.005
Actitudes
```

Creencias

tipo(rio) entero:2

posicion_x(rio) doble:0

posicion_y(rio) doble:2

distancia(rio) doble:0

limiteX0(sabana) doble:0

limiteX1(sabana) doble:100

limiteY0(sabana) doble:0

limiteY1(sabana) doble:100

distancia_minima(yo) doble:6

vista(yo) doble:10

estado_interno(yo) cadena:inicial

Intereses

miedo difuso superior:ligeramente inferior:nada

alegria difuso superior:absolutamente inferior:ligeramente

sorpresa difuso superior:absolutamente inferior:nada

sed difuso superior:medianamente inferior:ligeramente

hambre difuso superior:medianamente inferior:ligeramente

cansancio difuso superior:medianamente inferior:nada

RelacionesCreencias

influyente:valor influida:miedo disminuye difuso:ligeramente

influyente:valor influida:sorpresa disminuye difuso:ligeramente

PersoActitudes

influyente:valor influida:temor disminuye difuso:ligeramente

ActitudEmociones

influyente:temor influida:miedo aumenta difuso:bastante

influyente:temor influida:alegria disminuye difuso:medianamente

influyente:temor influida:sorpresa aumenta difuso:ligeramente

CreenciasIntereses

influyente:valor influida:miedo aumenta superior difuso:ligeramente

influyente:fortaleza influida:miedo aumenta superior difuso:ligeramente

influyente:fortaleza influida:sed aumenta superior difuso:ligeramente

influyente:fortaleza influida:hambre aumenta superior difuso:ligeramente

influyente:fortaleza influida:cansancio aumenta superior difuso:ligeramente

EstadoEmociones

influyente:sed influida:alegria disminuye difuso:medianamente

influyente:hambre influida:alegria disminuye difuso:medianamente

influyente:cansancio influida:alegria disminuye difuso:medianamente

B.2.2. Fichero de Acciones

beber

enviarConfirmacion

enviarRechazo

enviarResultado

terminaSolicitud

tratarSolicitud

enviarFallo

tratarServicio

B.2.3. Fichero de Reglas

regla

tipo() = entero:2

```
distancia() = doble:0
sed(yo) > umbral:superior
finantecedente
prioridad:5
accion:beber
Parametros
Parametro doble:3
finregla
```

B.2.4. Fichero de Reglas Metas

```
reglameta
estado_comunicacion() = cadena:analizando_solicitud
finantecedente
prioridad:6
caducidad:10000000
estado_comunicacion() = cadena:inicial
finmeta
```

Apéndice C

DOMINIO JSHOP2 DE SABANA 3D

En este último apéndice, se incluyen los ficheros de *Dominio* y *Problema* del Planificador JSHOP2 (ver sección 3.6).

El fichero de Dominio es el que proporciona el conjunto operadores, reglas, metas y certezas del mundo sobre el que el motor de JSHOP2 elabora los planes:

- Los operadores son las acciones a realizar, es decir, un plan se compone de un conjunto de operadores. En nuestro caso se incluyen los operadores *MOVER, BEBER,* ... que son las acciones que pueden llevar a cabo las cebras.

- Las certezas son predicados que son ciertos en el dominio.

- Las reglas son los razonamientos que dan lugar a la inclusión de nuevas certezas u operadores en función de otras certezas o de las metas definidas.

- Las metas son los objetivos a conseguir por el plan.

El fichero de Problema es el que determina el estado actual del mundo en el momento en que se lanza el planificador. Incluye el conjunto de certezas en ese momento junto con las metas a conseguir por el agente.

Ambos se incluyen para ilustrar el formato de la entrada requerida por el Planificador y así comprender la relación entre ésta y las Creencias y Metas de los agentes COGNITIVA

C.1. Fichero del Dominio *Zebras*

```
(defdomain zebras
; OPERADOR MOVER
(
(:operator (!mover ?animal ?x_actual ?y_actual ?x_nuevo ?y_nuevo ?direccion)
(
(animal ?animal)
(posicion ?animal ?x_actual ?y_actual)
)
(
(posicion ?animal ?x_actual ?y_actual)
)
(
(posicion ?animal ?x_nuevo ?y_nuevo)
)
)
(:operator (!beber ?animal)
(
(goal-bebiendo ?animal)
(posicion ?animal ?x_actual ?y_actual)
(agua ?animal ?x_actual ?y_actual)
)
(
)
(
(bebido ?animal)
```

```
)
)
(:operator (!comer ?animal)
(
(goal-comiendo ?animal)
(posicion ?animal ?x_actual ?y_actual)
(comida ?animal ?x_actual ?y_actual)
)
(
)
(
(comido ?animal)
)
)
(:operator (!guarecerse ?animal)
(
(goal-guarecido ?animal)
(posicion ?animal ?x_actual ?y_actual)
(guarida ?animal ?x_actual ?y_actual)
)
(
)
(
(guarecido ?animal)
)
)
(:operator (!bebiendo ?animal)
```

```
(
; (goal (bebiendo ?animal))
)
(
)
(
; (iniciado_beber ?animal)
)
)
(:operator (!fin-bebiendo ?animal)
(
; (iniciado_beber ?animal)
)
(
; (iniciado_beber ?animal)
)
(
)
)
(:operator (!comiendo ?animal)
(
)
(
)
(
)
)
```

```
(:operator (!fin-comiendo ?animal)
(
)
(
)
(
)
)
(:operator (!moviendo ?animal)
(
)
(
)
(
)
)
(:operator (!fin-moviendo ?animal)
(
)
(
)
(
)
)
(:operator (!aumentar-umbral-miedo ?animal)
(
)
```

```
(
)
(
)
)
(:operator (!decrementar-umbral-miedo ?animal)
(
)
(
)
(
)
)
(:operator (!tratar-solicitud ?animal)
(
)
(
)
(
)
)
(:operator (!solicitar ?animal)
(
)
(
)
(
```

```
)
)
(:method (deseo)
(
(goal-bebiendo ?animal)
)
(
(!bebiendo ?animal)
(cumplir_deseo ?animal)
(!fin-bebiendo ?animal)
(deseo)
)
(
(goal-comiendo ?animal)
)
(
(!comiendo ?animal)
(cumplir_deseo ?animal)
(!fin-comiendo ?animal)
(deseo)
)
(
(goal-guarecido ?animal)
)
(
(!moviendo ?animal)
(cumplir_deseo ?animal)
```

```
(!fin-moviendo ?animal)
(deseo)
)
(
(goal-solicitando ?animal)
)
(
(!solicitar ?animal)
)
(
(goal-sirviendo ?animal)
)
(
(!tratar-solicitud ?animal)
)
;Precondiciones estado meta
(
)
(
)
)
(:method (cumplir_deseo ?animal)
(
(goal-bebiendo ?animal)
(bebido ?animal)
)
(
```

```
(!!unassert(goal-bebiendo ?animal))
)
(
(goal-comiendo ?animal)
(comido ?animal)
)
(
(!!unassert(goal-comiendo ?animal))
)
(
(goal-guarecido ?animal)
(guarecido ?animal)
)
(
(!!unassert(goal-guarecido ?animal))
)
(
(goal-bebiendo ?animal)
(posicion ?animal ?x_actual ?y_actual)
(agua ?animal ?x_obj ?y_obj)
)
(
(!!assert((goal-posicion ?animal ?x_obj ?y_obj)))
(moverse)
(!beber ?animal)
)
(
```

```
(goal-comiendo ?animal)
(posicion ?animal ?x_actual ?y_actual)
(comida ?animal ?x_obj ?y_obj)
)
(
(!!assert((goal-posicion ?animal ?x_obj ?y_obj)))
(moverse)
(!comer ?animal)
)
(
(goal-guarecido ?animal)
(posicion ?animal ?x_actual ?y_actual)
(guarida ?animal ?x_obj ?y_obj)
)
(
(!!assert((goal-posicion ?animal ?x_obj ?y_obj)))
(moverse)
(!guarecerse ?animal)
)
nil
nil
)
(:method (moverse)
; Si hay leon
(
(no_hay_leon ?cebra)
)
```

```
(
; Mueves sin calcular distancias a leones
(mover_libre)
)
(
(animal ?animal)
(goal-posicion ?animal ?x_obj ?y_obj)
(posicion_sin_peligro ?x_obj ?y_obj)
)
(
;;Hay leones, pero son esquivables para llegar al rio
(mover_entre_leones)
)
(
(animal ?animal)
)
(
;;León inevitable
(!aumentar-umbral-miedo ?animal)
(mover_libre)
(!decrementar-umbral-miedo ?animal)
)
)
(:method (mover_entre_leones)
(
(goal-posicion ?animal ?x_obj ?y_obj)
(posicion ?animal ?x_obj ?y_obj)
```

```
)
(
(!!unassert(goal-posicion ?animal ?x_obj ?y_obj))
)
(
(goal-posicion ?animal ?x_obj ?y_obj)
(posicion ?animal ?x_actual ?y_actual)
)
(
(movimientos-posibles ?x_actual ?y_actual)
(realizar-movimiento-leones ?animal ?x_actual ?y_actual ?x_obj ?y_obj)
(mover_entre_leones)
)
nil
nil
)
(:method (realizar-movimiento-leones ?animal ?x_actual ?y_actual ?x_obj ?y_obj)
(:sort-by ?distancia_euclidea
(
(adyacente ?x_nuevo ?y_nuevo ?direccion)
(posicion_valida ?x_nuevo ?y_nuevo)
;; Condiciones sobre cercania leones
;(call funcionImprimir ?x_actual ?y_actual ?x_nuevo ?y_nuevo)
(posicion_sin_peligro ?x_nuevo ?y_nuevo)
(distancia ?x_nuevo ?y_nuevo ?x_obj ?y_obj ?distancia_euclidea)
)
)
```

```
(
(!mover ?animal ?x_actual ?y_actual ?x_nuevo ?y_nuevo ?direccion)
(!!assert((obstaculo ?x_actual ?y_actual)))
)
)
(:- (posicion_sin_peligro ?x_nuevo ?y_nuevo)
(
(forall (?leon) (leon ?leon) (distancia_fuera_rango_leon ?x_nuevo ?y_nuevo ?leon))
)
)
(:- (distancia_fuera_rango_leon ?x_nuevo ?y_nuevo ?leon)
(
(posicion ?leon ?x_leon ?y_leon)
(distancia-minima ?d)
;(call funcionImprimir (call funcionDistancia ?x_nuevo ?y_nuevo ?x_leon ?y_leon))
(distancia ?x_nuevo ?y_nuevo ?x_leon ?y_leon ?distancia_euclidea)
(call <= ?d ?distancia_euclidea)
)
)
(:method (mover_libre)
(
(goal-posicion ?animal ?x_obj ?y_obj)
(posicion ?animal ?x_obj ?y_obj)
)
(
(!!unassert(goal-posicion ?animal ?x_obj ?y_obj))
)
```

```
(
(goal-posicion ?animal ?x_obj ?y_obj)
(posicion ?animal ?x_actual ?y_actual)
)
(
(movimientos-posibles ?x_actual ?y_actual)
(realizar-movimiento ?animal ?x_actual ?y_actual ?x_obj ?y_obj)
(mover_libre)
)
nil
nil
)
(:method (realizar-movimiento ?animal ?x_actual ?y_actual ?x_obj ?y_obj)
(:sort-by ?distancia_euclidea
(
(adyacente ?x_nuevo ?y_nuevo ?direccion)
(posicion_valida ?x_nuevo ?y_nuevo)
(distancia ?x_nuevo ?y_nuevo ?x_obj ?y_obj ?distancia_euclidea)
)
)
(
(!mover ?animal ?x_actual ?y_actual ?x_nuevo ?y_nuevo ?direccion)
)
)
(:method (movimientos-posibles ?x_actual ?y_actual)
(
(assign ?y_mas1 (call + ?y_actual 1))
```

```
(assign ?y_menos1 (call - ?y_actual 1))
(assign ?x_mas1 (call + ?x_actual 1))
(assign ?x_menos1 (call - ?x_actual 1))
)
(
(borrar-adyacentes)
(!!assert
(
(adyacente ?x_actual ?y_mas1 0)
(adyacente ?x_actual ?y_menos1 4)
(adyacente ?x_mas1 ?y_actual 2)
(adyacente ?x_menos1 ?y_actual 6)
(adyacente ?x_mas1 ?y_mas1 1)
(adyacente ?x_menos1 ?y_menos1 5)
(adyacente ?x_mas1 ?y_menos1 3)
(adyacente ?x_menos1 ?y_mas1 7)
)
)
;(print-current-state)
)
)
(:method (borrar-adyacentes)
(
(adyacente ?x ?y ?z)
)
(
(!!unassert(adyacente ?x ?y ?z))
```

```
(borrar-adyacentes)
)
nil
nil
)
(:- (posicion_valida ?x_nuevo ?y_nuevo)
(
(not (obstaculo ?x_nuevo ?y_nuevo))
(call <= 0 ?x_nuevo)
(call <= 0 ?y_nuevo)
(forall (?leon) (leon ?leon) (not (posicion ?leon ?x_nuevo ?y_nuevo)))
)
)
(:- (distancia ?x_actual ?y_actual ?x_nuevo ?y_nuevo ?distancia_euclidea)
(
(assign ?resta1 (call - ?x_actual ?x_nuevo))
(absoluto ?resta1 ?primero)
(assign ?resta2 (call - ?y_actual ?y_nuevo))
(absoluto ?resta2 ?segundo)
(assign ?distancia_euclidea (call + ?primero ?segundo))
)
)
(:- (absoluto ?num ?abs)
(
(or
(and(call < ?num 0)
(assign ?abs (call - 0 ?num))
```

```
)
(and(call >= ?num 0)
(assign ?abs ?num)
)
)
)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;METODOS AUXILIARES
;Estos operadores, nos anaden y eliminan predicados del estado del mundo
(:operator (!!assert ?atoms) () () ?atoms 0)
(:operator (!!unassert ?x) () (?x) () 0)
;Este metodo es el metodo inicial al que llamamos desde la definicion del problema
(:method (achieve-goals ?goals)
()
((assert-goals ?goals nil)
; (print-current-state)
(deseo)
)
)
; Metodos para insertar los objetivos en el estado del mundo
(:method (assert-goals (?first . ?rest) ?atoms)
()
((assert-goals ?rest ((goal ?first) . ?atoms))))
(:method (assert-goals nil ?atoms)
()
((!!assert ?atoms)))
```

```
; axiomas auxiliares
(:- (iguales ?x ?x) nil)
(:- (iguales ?x ?x ?x) nil)
(:- (diferentes ?x ?y) ((not (iguales ?x ?y))))
(:method (swap ?x ?y)
((have ?x) (not (have ?y)))
((!drop ?x) (!pickup ?y))
((have ?y) (not (have ?x)))
((!drop ?y) (!pickup ?x)))))
)
)
```

C.2. Fichero del Problema

```
(defproblem problem zebras
(
(animal cebrita)
(agua cebrita 0 33)
(comida cebrita 0 15)
(no_hay_leon cebrita)
(distancia-minima 6)
(posicion cebrita 0 2)
(goal-solicitando cebrita)
)
(
(deseo)
)
```

```
)
```